

ReProCS: A Missing Link between Recursive Robust PCA and Recursive Sparse Recovery in Large but Correlated Noise

Chenlu Qiu and Namrata Vaswani

Abstract—This work studies the recursive robust principal components’ analysis (PCA) problem. Here, “robust” refers to robustness to both independent and correlated sparse outliers, although we focus on the latter. A key application where this problem occurs is in video surveillance where the goal is to separate a slowly changing background from moving foreground objects on-the-fly. The background sequence is well modeled as lying in a low dimensional subspace, that can gradually change over time, while the moving foreground objects constitute the correlated sparse outliers. In this and many other applications, the foreground is an outlier for PCA but is actually the “signal of interest” for the application; where as the background is the corruption or noise. Thus our problem can also be interpreted as one of recursively recovering a time sequence of sparse signals in the presence of large but spatially correlated noise.

This work has two key contributions. First, we provide a new way of looking at this problem and show how a key part of our solution strategy involves solving a noisy compressive sensing (CS) problem. Second, we show how we can utilize the correlation of the outliers to our advantage in order to even deal with very large support sized outliers. The main idea is as follows. The correlation model applied to the previous support estimate helps predict the current support. This prediction serves as “partial support knowledge” for solving the modified-CS problem instead of CS. The support estimate of the modified-CS reconstruction is, in turn, used to update the correlation model parameters using a Kalman filter (or any adaptive filter). We call the resulting approach “support-predicted modified-CS”.

I. INTRODUCTION

Most high dimensional data often approximately lies in a lower dimensional subspace. Principal Components’ Analysis (PCA) is a widely used dimension reduction technique that finds a small number of orthogonal basis vectors (principal components), along which most of the variability of the dataset lies. To be precise, for a given dimension, r , PCA finds the r -dimensional subspace that minimizes the mean squared error between data vectors and their r -dimensional projections [3]. The subspace spanned by the principal components (PCs) is called the principal components’ space (PC space). Often, for time series data, the PC space changes gradually over time. Updating the PC space recursively as more data comes in, without re-solving the entire PCA problem, is referred to as recursive PCA [4].

Notice that to find an r dimensional PC space, one needs at least r data vectors, usually more. Thus, even for recursive

PCA, the initial step needs to be a batch one or the initial PC space needs to be pre-specified.

It is well known that PCA is very sensitive to outliers. Computing the PCs in the presence of outliers is called robust PCA [5], [6], [7], [8]. Doing recursive PCA in the presence of outliers is referred to as recursive robust PCA [4], [9], [10]. “Outlier” is a loosely defined term that usually refers to any corruption that is not small compared to the true data vector and that occurs only occasionally. As suggested in [11], [7], an outlier can be very nicely modeled as a sparse vector, i.e. a vector whose most elements are zero, while the few that are nonzero can have any magnitude. We will use this definition in this paper. In [11], [7], the outlier is modeled as being spatially and temporally independent. In most real applications, the time at which the outliers begin to occur is often random and independent of all past times. But once outliers begin to occur, for some time after that they are both spatially and temporally correlated. In this work, we focus on this case, i.e. on *recursive robust PCA that is robust to correlated sparse outliers*.

A key application where the robust PCA problem occurs is in video surveillance where the goal is to separate a slowly changing background from moving foreground objects [6], [7]. If we stack each video frame as a column vector, then the background is well modeled as lying in a low dimensional subspace that may gradually change over time, while the moving foreground objects constitute the sparse outliers [11][7] which change in a correlated fashion over time. *We will use this as the motivating application in this work.* Other important applications include sensor networks based detection and tracking of abnormal events such as forest fires or oil spills; or online detection of brain activation patterns from fMRI sequences (the “active” part of the brain can be interpreted as a correlated sparse outlier). Clearly, in all these cases, one would need a real-time and fast solution and hence a recursive robust PCA solution is desirable.

The moving objects or the brain active regions or the oil spill region may be “outliers” for the PCA problem, but in most cases, these are actually the “signals of interest” whereas the background image is the noise. Also, all the above “signals of interest” are sparse vectors that change in a correlated fashion over time. Thus, this problem can also be re-interpreted as one of *recursively recovering a time sequence of correlated sparse signals, S_t , in the presence of large but “low rank” noise, L_t .*

Definition 1: The term “low rank” vector means that the $n \times \tau$ matrix $L := [L_{t-\tau+1}, \dots, L_t]$ has low-rank, i.e. its rank is less than $\min(n, \tau)$ for τ large enough. This would happen

if L_t is correlated enough to have a low rank covariance matrix and this matrix changes slowly over time. We make this precise below in Sec. I-A.

A. Problem Definition

Our problem can be defined as follows. The measurement vector at time, t , M_t , is an $n \times 1$ vector that satisfies

$$M_t = L_t + S_t \quad (1)$$

where S_t is a sparse vector, with support set denoted by T_t , and L_t is a dense (non sparse) but “low rank” vector. The support set of S_t , T_t , can be correlated over time and space.

Suppose, we have a good estimate of the initial PC matrix, $\hat{P}_0 \approx P_0$. For $t > 0$, our goal is to recursively keep estimating the sparse part, S_t , at each time, and to keep updating \hat{P}_t every-so-often, by collecting the recent estimates of $L_t = M_t - S_t$.

To make things precise, we assume that L_t satisfies

$$L_t = U x_t$$

where U is an *unknown* orthonormal matrix and x_t is an $n \times 1$ sparse vector whose support changes every-so-often and whose elements are spatially uncorrelated. Let N_t denote the support set of x_t . We assume that N_t is piecewise constant with time. Thus, the columns of the sub-matrix, $P_t := (U)_{N_t}$, span the low dimensional subspace in which the current set of L_t 's lie and $L_t = P_t(x_t)_{N_t}$. We refer to P_t as the principal components' (PC) matrix. Clearly, this is also piecewise constant with time.

Every d time units, there are k additions to the set N_t , or, equivalently, k directions get added to P_t . When a new direction gets added, the magnitude of x_t along it is initially small but gradually increases to a larger stable value. Also, the values of x_t along k existing directions gradually decays down to zero, i.e. the corresponding directions get slowly removed from P_t . We provide a generative model that satisfies these assumptions in the Appendix. It models x_t (and hence L_t) as being piecewise stationary with short nonstationary transients between the pieces that occur whenever the support of x_t changes.

In the video problem, the observed image is an *overlay* of the foreground and the background image. Denote the image, background image and foreground image written as a 1D vector by M_t , L_t and O_t respectively. Let T_t denote the support of O_t and let T_t^c denote the complement set of T_t . By overlay, we mean that $(M_t)_{T_t} = (O_t)_{T_t}$ and $(M_t)_{T_t^c} = (L_t)_{T_t^c}$. We can rewrite this in the form of (1) by defining S_t as

$$(S_t)_i = \begin{cases} (O_t - L_t)_i & i \in T_t \\ 0 & i \in T_t^c \end{cases} \quad (2)$$

If we refer to the above problem as *recursive robust PCA*, then the low dimensional vector, L_t , is the “signal of interest” while the correlated sparse vector, S_t , is the corruption (outlier). On the other hand, if we refer to it as *recursive sparse recovery in large but spatially correlated (low rank) noise*, L_t , then the correlated sparse vector, S_t , is the signal of interest, while L_t is the corruption (large but low rank noise).

An example of noise L_t being much larger than signal S_t is shown in Fig. 1(c). In the rest of the paper, we just refer to S_t as the *sparse part* and L_t as the *low rank part*.

In this work, we focus on the $M_t = S_t + L_t$ case, but our proposed ideas will also apply if

$$M_t = \Psi S_t + L_t \quad (3)$$

where Ψ can be a fat matrix. This would be the standard sparse recovery problem from a reduced number of measurements but with the difference that the measurements are corrupted by very large noise. Also, this allows the signal to be sparse in some other basis other than the canonical basis. With the exception of [11], which can handle another kind of very large but “structured” noise (the noise or outlier needs to be sparse), almost all other existing sparse recovery as well as recursive sparse recovery approaches for time sequences only work with small noise [12], [13], [14], [15], [16], [17], [18], [19], [20] (surely none of these will work if the noise is significantly larger than the sparse part).

B. Contributions and Paper Organization

Our first contribution is to show how our problem can be reformulated as a sparse recovery / compressive sensing (CS) plus recursive PCA problem. We call our solution *Recursive Projected Compressive Sensing (ReProCS)*. Its block diagram is shown in Fig. 2(a). The key idea is as follows. Assume that the current PC matrix P_t has been accurately estimated, i.e. we are given $\hat{P}_t \approx P_t$. We project the measurement vector, M_t , into the space perpendicular to \hat{P}_t to get an $n - r$ dimensional projected measurement vector y_t . Here, r is the rank of \hat{P}_t . This projection nullifies most of the contribution of L_t . It is assumed that this projection does not nullify any nonzero component of the sparse vectors, S_t . Recovering S_t from y_t now becomes a traditional noisy CS [14], [21], [22] problem. The recovered S_t can be used to estimate L_t which can then be used to update \hat{P}_t every-so-often (recursive PCA). *In this work we misuse terminology a little and use “compressive sensing” or “CS” to refer to the ℓ_1 minimization problem.*

Our second key contribution is to show how we can utilize the correlation of the sparse part, S_t , to our advantage to successfully recover S_t even when its support size, $|T_t|$, increases for a given rank r of P_t and as a result the number of projected “measurements” available for the CS step, $n - r$, become too small for CS to work. The correlated sparse outliers (e.g. moving foreground objects in video) can be interpreted as sparse signal sequences whose *support change over time is either slow; or quite often is not slow, but is still correlated, e.g. the support can “move” or “expand” or change according to some model, over time.* By using a model on how the objects move or deform, or other models on how the support changes, it is possible to obtain a support prediction that serves as an accurate “partial support knowledge”. We then tap into our recent work on modified-CS (modCS) which solves the sparse recovery problem with much fewer measurements when reliable support knowledge is available [23]. The support estimate of the modCS reconstruction can then be used to update the correlation model parameters using a Kalman filter

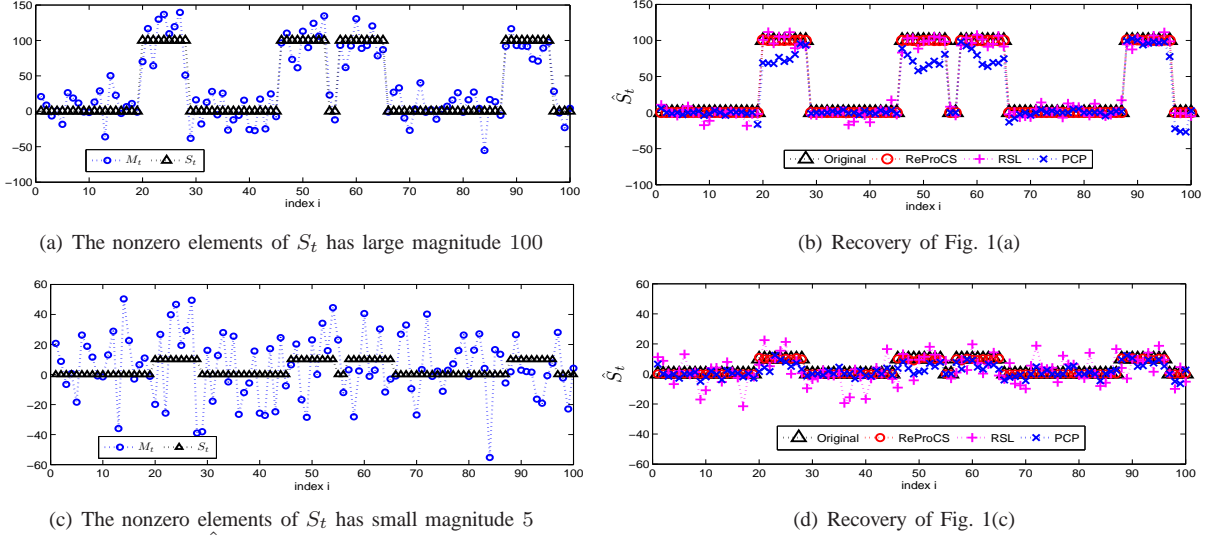


Fig. 1: We plot M_t , S_t , and \hat{S}_t at $t = t_0 + 5$ for the experiment described in Sec. VI-A. The sparse part S_t has large magnitude in the top row and small magnitude in the bottom row. As a result, RSL [6] works in the first case but not the second one. The support sets of S_t 's are large and generated in a correlated fashion and so the sparse matrix $S = [S_1, \dots, S_{t_0+200}]$ also has low rank (rank = 45%). This is the reason PCP [7] does not work in both cases (also see Table I, PCP does work in case of small magnitude but small support S_t 's).

or any other adaptive filter. We call this *support predicted modified-CS (supp-pred-modCS)*. Its block diagram shown in Fig. 2(b).

This paper is organized as follows. We discuss related work next and define notation in Sec. II. The ReProCS idea is described in Sec. III. The supp-pred-modCS idea and the overall ReProCS(modCS) approach is developed in Sec. IV. We also discuss its stability over time. In Sec. V, we explain the recursive PCA algorithm which is based on the idea of [4]. Simulation experiments and partly-real experiments (real background images with simulated foreground sparse image) evaluating the performance of ReProCS and ReProCS(modCS) and comparing them with other state-of-art methods – two recursive robust PCA methods, [10] and [4], two batch robust PCA methods, principal components' pursuit (PCP) [7] and robust subspace learning (RSL) [6], and with simple thresholding based background subtraction (for the video case) are given in Sec. VI. Conclusions, limitations and future work are given in Sec. VII. *Sections III-B, IV-C and V can be skipped on a quick reading. We mark them with a **.*

For ease of review, both our related work discussion and our experiments' section is very detailed. Some of this material can be shortened/removed eventually.

C. Related Work

There has been a large amount of work on robust PCA, e.g. [5], [6], and recursive robust PCA e.g. [4], [9], [10]. In most of these works, either the locations of the missing/corrupted data points are assumed known [4] (not a practical assumption); or they first detect the corrupted data points and then replace their values using nearby values [9]; or weight each data point in proportion to its reliability (thus soft-detecting and down-weighting the likely outliers) [6], [10]; or just remove the entire outlier vector. Approaches like [4] can be adapted to the case where the missing/corrupted data points are unknown by using the outlier detection approach from other works,

e.g. from [10] (we refer to the resulting method as adapted-[4]). Detecting or soft-detecting outliers (sparse part S_t) as in [9], [6], [10] is easy to do when their magnitude is large compared to that of L_t , but not when it is smaller, e.g. see Fig 1 and Table I. When the signal of interest is S_t (the case of recursive sparse recovery in large but low rank noise), the most difficult situation is when nonzero elements of S_t have small magnitude compared to those of L_t . Moreover, as we explain in Sec. VI, approaches such as [10] or adapted-[4] also cannot work in case of too many outliers (large support size of S_t), e.g. see Table I. But ReProCS is able to successfully recover both small magnitude and fairly large support-sized S_t 's because it operates by first approximately nullifying L_t and then recovering S_t by solving a noisy CS problem, that enforces sparsity of the S_t 's.

In a series of recent works [7], [8], [24], an elegant solution has been proposed, that does not require a two step outlier location detection/correction process and also does not throw out the entire vector. It redefines batch robust PCA as a problem of separating a low rank matrix, $L := [L_1, \dots, L_t]$, from a sparse matrix, $S := [S_1, \dots, S_t]$, using the data matrix, $M := [M_1, \dots, M_t] = L + S$. It was shown in [7] that one can recover L and S exactly by solving

$$\min_{L, S} \|L\|_* + \lambda \|S\|_1 \text{ subject to } L + S = M \quad (4)$$

where $\|L\|_*$ is the sum of singular values of L while $\|S\|_1$ is the ℓ_1 norm of S seen as a long vector, provided that

- the singular vectors of L are spread out enough (not sparse),
- the support and signs of S are uniformly random (thus it is not low rank) and
- the rank of L is sufficiently small for a given sparsity of S .

This was called *Principal Components' Pursuit (PCP)*. While PCP is an elegant idea, it has three important limitations.

- Most importantly, PCP relies on the fact that the matrix $S := [S_1, \dots, S_t]$ is sparse but full rank. But when the S_t 's are correlated and have a large support size, S will also often be low rank. This is particularly easy to see for the case where the support sets, T_t , have large overlaps over time. For example, for the case of Fig 1, the rank of S is only 45 (while $n = 100$). A low rank S makes it impossible for PCP to separate S from L and hence neither is recovered correctly. Also see Fig. 3.
- A related issue is that PCP requires the rank of L to be quite small for a given support size of S_t , $|T_t|$, e.g. see [7, Table 1]. But, real videos can have a lot of background variations, e.g. see Sec. VI-D, causing the rank of L to be as much as 20% of the image size. Also, the foreground can have large sized and multiple moving objects, making $|T_t|$ also quite large.
- In many applications, e.g. in surveillance, one would like to obtain the estimates on-the-fly and quickly as a new frame comes in, rather than in a batch fashion.

Our proposed algorithms, ReProCS and ReProCS(modCS), address these drawbacks. Unlike PCP, (a) ReProCS does not need the S_t 's to be uncorrelated, and (b) it can recover S_t 's with larger support sizes [see Figs. 1, 3 and Table II]. Moreover, our second solution, ReProCS(modCS), utilizes the correlation of the S_t 's to its advantage in order to successfully recover S_t 's with significantly larger support sizes than what ReProCS can [see Figs 4, 5]. Finally, both ReProCS and ReProCS(modCS) are recursive methods and hence they provide real-times estimates. Of course, our work and in fact any recursive robust PCA method, e.g. [10], [4], does need an initial estimate of the PC matrix which PCP or other batch methods, e.g. [6], do not need. In practice, as we explain later in Sec. V, this is usually easy to obtain.

The static version of our problem is also somewhat related to that of [25], [26], [27] in that all of these also try to cancel the “low rank” part by projecting the original data vector into the perpendicular space of the tall matrix that spans the “low rank” part. But the big difference is that in all these, this matrix is *known*. In our problem P_t is unknown and can change with time, and we also show how we can utilize the correlation of the outliers to our advantage.

If U itself were known, then at any given time, our problem would be similar to the dense error correction problem studied in [11], [28]. Of course the reason we need PCA is because U is *unknown and cannot even be estimated*. Only, the PCs, $P_t := (U)_{N_t}$ can be estimated.

Some recent work which actually is completely different from our current work, but may appear related (since it also uses CS and PCA or eigenvalue decomposition) includes [29] and [30].

II. NOTATION

The set operations \cup , \cap and \setminus have the usual meanings. For any set $T \subset \{1, \dots, n\}$, T^c denotes its complement, i.e., $T^c := \{i \in [1, \dots, n] : i \notin T\}$, and $|T|$ denotes its cardinality, i.e., the number of elements in T . But $|a|$ where a is a real number denotes the magnitude of a .

For a vector v , v_i denotes the i th entry of v and v_T denotes a vector consisting of the entries of v indexed by T . We use $\|v\|_p$ to denote the ℓ_p norm of v . The support of v , $\text{supp}(v)$, is the set of indices at which v is nonzero, $\text{supp}(v) := \{i : v_i \neq 0\}$. We say that v is s -sparse if $|\text{supp}(v)| \leq s$. Sorting $|v_i|$ in descending order, we define the $p\%$ -energy set of v as $T_p := \{|v_i| \geq \kappa\}$ where κ is the largest value of $|v_i|$ such that $\|v_{T_p}\|_2^2 \geq p\% \|v\|_2^2$, i.e., v_{T_p} contains the significantly nonzero elements of v .

For a matrix A , A_i denotes the i th column of A and A_T denotes a matrix composed of the columns of A indexed by T . We use A_{T_1, T_2} to denote a submatrix of A consisting of the rows indexed by T_1 and columns indexed by T_2 . We use A' to denote its transpose, and A^\dagger to denote its pseudoinverse. For a tall matrix A , $A^\dagger = (A'A)^{-1}A'$. The Frobenius norm of matrix A is denoted by $\|A\|_F$, i.e., $\|A\|_F := \sqrt{\sum_i \sum_j |A_{i,j}|^2}$.

For a tall matrix P , we use $\text{span}(P)$ to denote the subspace spanned by the column vectors of P .

For a diagonal matrix D , D_T denotes a submatrix of D consisting of the rows and columns indexed by T . In other words, D_T is a diagonal matrix with $(D_T)_{j,j} = (D)_{T_j, T_j}$. Also, $\text{diag}(D)$ denotes the vector composed of the diagonal elements of D , i.e., $(\text{diag}(D))_i = D_{i,i}$.

We use \emptyset to denote an empty set or an empty matrix.

III. RECURSIVE PROJECTED COMPRESSIVE SENSING

We explain the Recursive Projected Compressive Sensing (ReProCS) algorithm below. In Sec. III-B, we discuss the implicit assumptions required for it.

A. Recursive Projected Compressive Sensing (ReProCS) algorithm

Let \hat{P}_t be an estimate of the PC matrix P_t at time t and let $\hat{P}_{t,\perp}$ be an orthogonal complement of \hat{P}_t . The column space of $\hat{P}_{t,\perp}$ is the null space of \hat{P}_t' , and hence $\hat{P}_{t,\perp}$ is not unique. Using \hat{P}_t and $\hat{P}_{t,\perp}$, we can rewrite $L_t = \hat{P}_t \alpha_t + \hat{P}_{t,\perp} \beta_t$ and hence M_t as

$$M_t = \hat{P}_t \alpha_t + \hat{P}_{t,\perp} \beta_t + S_t$$

where $\alpha_t := \hat{P}_t' L_t$ is the projection of L_t into the subspace spanned by \hat{P}_t ; and $\beta_t := (\hat{P}_{t,\perp})' L_t$ is the projection of L_t into the subspace spanned by $\hat{P}_{t,\perp}$.

To approximately nullify the low rank part, L_t , we can project the data vector, M_t , into the space spanned by $\hat{P}_{t,\perp}$, i.e. compute

$$y_t := A_t M_t, \text{ where } A_t := (\hat{P}_{t,\perp})'. \quad (5)$$

The dimension of the projected data vector, y_t , reduces to $n-r$ where $r := \text{rank}(\hat{P}_t)$. Notice that

$$y_t = A_t S_t + \beta_t, \text{ where } \beta_t := A_t L_t = A_t P_t (x_t)_{N_t} \quad (6)$$

If $\hat{P}_t \approx P_t$, then $A_t P_t \approx 0$, i.e. this nullifies most of the contribution of the low rank part, so that β_t can be interpreted as small “noise”. Finding the n -dimensional sparse vector, S_t , from this $n-r$ dimensional projected data vector, y_t , now becomes the traditional noisy sparse reconstruction / compressive sensing (CS) [14], [21], [22] problem with the

Algorithm 1 ReProCS

At $t = t_0$, suppose a good estimate of PC matrix, \hat{P}_{t_0} is available from training data. For $t > t_0$, do the following:

- 1) Let $A_t \leftarrow A_{t-1}$. Obtain y_t by (5).
- 2) Estimate S_t by solving (7) with $\epsilon = \|\hat{P}'_{t,\perp} \hat{L}_{t-1}\|_2$.
- 3) Support thresholding and least square estimation: do (10) and (11).
- 4) Estimate $\hat{L}_t = M_t - \hat{S}_t$.
- 5) Update \hat{P}_t using Algorithm 4 which is based on [4]. Update $A_t := (\hat{P}_{t,\perp})'$.
- 6) Increment t by 1 and go to step 1).

In practice, one can replace (7) by (8) with $\epsilon = \|(I - \hat{P}_t \hat{P}'_t) \hat{L}_{t-1}\|_2$.

“projected noise” β_t resulting from the error in estimating P_t . As long as $\|\beta_t\|_2$ is small and A_t does not nullify any nonzero elements of S_t , we can recover S_t by solving

$$\min_s \|s\|_1 \text{ subject to } \|y_t - A_t s\|_2 \leq \epsilon \quad (7)$$

with ϵ chosen proportional to the “noise” level, $\|\beta_t\|_2$. Denote its output by \hat{S}_t . In practice, for large scale problems where n is large, a less computationally and memory intensive way than (7) is to solve

$$\min_s \|s\|_1 \text{ subject to } \|(I - \hat{P}_t \hat{P}'_t)(M_t - s)\|_2 \leq \epsilon \quad (8)$$

This is exactly equivalent to (7) because $\hat{P}_{t,\perp}$ is an orthonormal complement of \hat{P}_t satisfying $\hat{P}_t \hat{P}'_t + \hat{P}_{t,\perp} \hat{P}'_{t,\perp} = I$ and $\|\hat{P}'_{t,\perp} M_t\|_2 = \|\hat{P}_{t,\perp} \hat{P}'_{t,\perp} M_t\|_2$.

Using \hat{S}_t , we can then estimate

$$\hat{L}_t = M_t - \hat{S}_t \quad (9)$$

which can be used to recursively update the PC matrix estimate, \hat{P}_t , every-so-often to prevent the “noise” β_t from getting large (recursive PCA). We explain how to do this in Sec. V.

In (7), we need an appropriate parameter ϵ which should be proportional to the “noise” term $\|\beta_t\|_2$. We set ϵ adaptively as $\epsilon = \|\hat{\beta}_{t-1}\|_2 = \|\hat{P}'_{t-1,\perp} \hat{L}_{t-1}\|_2$ at time t .

If the constraint in (7) is too tight (ϵ is too small), it will give a solution with too many nonzero values. Moreover, as first explained in [31], the solution of (7) is always biased towards zero due to minimizing the ℓ_1 norm. To address both these issues, as suggested in [31], we can do support estimation followed by least squares (LS) estimation on the support, i.e., we can compute

$$\hat{T}_t = \{i : (\hat{S}_t)_i \geq \gamma\} \quad (10)$$

$$(\hat{S}_t)_{\hat{T}_t} = ((A_t)_{\hat{T}_t})^\dagger y_t, (\hat{S}_t)_{\hat{T}_t^c} = 0 \quad (11)$$

The low rank part, L_t , can then be estimated using this new \hat{S}_t and (9).

The block diagram of the above approach is shown in Fig. 2(a) and the stepwise algorithm is given Algorithm 1.

B. Implicit Requirements**

Suppose that $|T_t| \leq s$ for all t , i.e. all S_t ’s are s sparse. Clearly, a necessary condition for the CS step of ReProCS, i.e. (7), to recover the support of all S_t ’s correctly is that $A_t = (\hat{P}_{t,\perp})'$ does not nullify any s -sparse vector, and thus does not nullify any nonzero part of any S_t . Or, equivalently, no s -sparse vector belongs to $\text{span}(\hat{P}_t)$. We can use the null space property (NSP) [32] to show that a slightly stronger version of this requirement also serves as a sufficient condition, at least in the noise-free case.

Consider (7). Assume that $\hat{P}_t = P_t$ so that the projected noise is indeed zero, i.e. $\beta_t = 0$. In this case, (7) with $\epsilon = 0$ will exactly recover any s -sparse S_t if the following holds with a $\theta < 1$:

$$\|(\eta)_T\|_1 \leq \theta \|(\eta)_{T^c}\|_1$$

for all sets T with $|T| \leq s$ and for all $\eta \in \text{null}(A_t)$ [32], [17]. Here $\text{null}(A_t) := \{\eta : A_t \eta = 0\}$ refers to the null space of A_t . In words, we need that all s -sparse or s -approximately-sparse vectors (i.e. vectors for which $\|(\eta)_{T^c}\|_1 \leq \|(\eta)_T\|_1$ for some set T with $|T| \leq s$) do not lie in $\text{null}(A_t)$. But $\text{null}(A_t) = \text{span}(\hat{P}_t)$. Thus, a sufficient condition for ReProCS to exactly recover S_t in the noise-free case ($\beta_t = 0$ and $\epsilon = 0$ in (7)) is that no s -sparse or s -approximately-sparse vector lies in $\text{span}(\hat{P}_t)$.

We expect that this should hold when the columns of P_t are spread out enough (not sparse), which, in turn, should hold if the changes of the background, L_t , are not localized in one or more small regions, e.g. due to water waves’ motion in the video application. We show an example in Sec. VI-D. This observation will be analyzed in future work.

Remark 1: We should note that we can also get a sufficient condition using the restricted isometry property (RIP) [25], [33] and in fact it would hold even in the noisy case, but it is not as illustrative. Let δ_s be the s -RIP constant [25] for the matrix A_t . If the projected noise β_t satisfies $\|\beta_t\|_2 \leq \epsilon$ and if $\delta_{2s} < \sqrt{2} - 1$, the reconstruction error is bounded by a constant times ϵ , i.e. $\|\hat{S}_t - S_t\|_2 \leq C(\delta_{2s})\epsilon$ [33].

IV. SUPPORT-PREDICTED MODIFIED-CS FOR REPROCS

ReProCS needs to recover a $|T_t|$ -sparse vector, S_t , from the projected data vector y_t by solving a noisy CS problem (7). The number of projected measurements available for the CS step (7) is $n - r$, where $r = \text{rank}(\hat{P}_t)$. For a given r , if the support size of S_t , $|T_t|$, increases, or if r increases for a given sparsity level, $|T_t|$, the ratio $\frac{|T_t|}{n-r}$ may become too large for CS to estimate S_t accurately. In this section we show how to utilize the correlated support changes of the S_t ’s to accurately recover them even when the ratio $\frac{|T_t|}{n-r}$ is too large for CS to work. We begin by first explaining the model on the support change of S_t . We then explain the *support-predicted modified-CS* algorithm, discuss why it is stable, and its extensions to more general cases. The complete ReProCS(modCS) algorithm is summarized in Algorithm 3.

A. Model on the support change of S_t

For explaining our ideas in a simple fashion, we consider a simple but realistic correlation model on S_t inspired by the

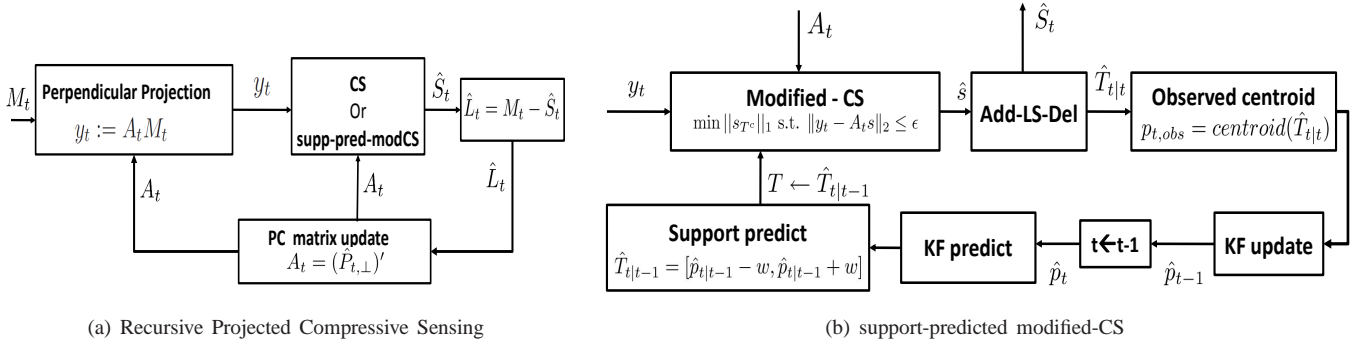


Fig. 2: Recursive Projected Compressive Sensing and support-predicted modified-CS

video application. We assume S_t is a 1D foreground image with one moving foreground object that satisfies a constant velocity model described below. The extension to 2D images with multiple moving objects or to other correlation models is simple and is explained in Sec. IV-D.

Let p_t be the location of the foreground object's centroid at time t , let v_t denote its velocity, and let w denote its width. Thus, its support is,

$$T_t = [p_t - w, p_t + w] \quad (12)$$

Let

$$g_t := \begin{bmatrix} p_t \\ v_t \end{bmatrix} \text{ and } G := \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

We assume the standard constant velocity model with small random acceleration on the object's motion [34, Example V.B.2], i.e.,

$$g_t = G g_{t-1} + \begin{bmatrix} 0 \\ n_t \end{bmatrix} \quad (13)$$

The modeling error (acceleration), n_t , is assumed to be bounded, with zero mean and variance Q .

B. Support-predicted modified-CS (supp-pred-modCS)

The main idea of supp-pred-modCS is as follows. We use the above model in a Kalman filter (KF) to track the object's motion over time. The KF predicted location of the object and its size tells us its predicted support at the current time. This is then used to solve the modified-CS (modCS) problem and obtain an updated support estimate. The centroid (or median) of this support estimate tells us the observed location of the object, which may be erroneous because our support estimate is not perfect. This then serves as the noisy observation for the KF update step to update the current location and velocity estimates. We now explain each of these four steps.

Predict Location:

Let $\hat{g}_{t|z} = [\hat{p}_{t|z} \ \hat{v}_{t|z}]'$ denote the estimate of g_t at time t given measurements up to and including at time z . Similar rule applies for $\hat{T}_{t|z}$. Let $\Sigma_{t|t-1}$, $\Sigma_{t|t}$ and K_t denote the prediction and updated error covariance matrices and the Kalman gain used by the KF. Compute

$$\hat{g}_{t|t-1} = G \hat{g}_{t-1|t-1}, \text{ and} \quad (14)$$

$$\Sigma_{t|t-1} = G \Sigma_{t-1|t-1} G' + \begin{bmatrix} 0 \\ Q \end{bmatrix} \quad (15)$$

Algorithm 2 Support-predicted modified-CS

- 1) Predict centroid by (14) and (15)
- 2) Predict support by (20)
- 3) Update support
 - Modified-CS: solve (21) with $T = \hat{T}_t$ and $\epsilon = \|\hat{P}'_{t,\perp} \hat{L}_{t-1}\|_2$.
 - Add-LS-Del procedure:

$$T_{\text{add}} = T \cup \{i \in T^c : |(\hat{s})_i| > \alpha_{\text{add}}\} \quad (16)$$

$$(\hat{s})_{T_{\text{add}}} = ((A_t)_{T_{\text{add}}})^\dagger y_t, \ (\hat{s})_{T_{\text{add}}^c} = 0 \quad (17)$$

$$\hat{T}_{t|t} = T_{\text{add}} \setminus \{i \in T_{\text{add}} : |(\hat{s})_i| < \alpha_{\text{del}}\} \quad (18)$$

$$(\hat{S}_t)_{\hat{T}_{t|t}} = ((A_t)_{\hat{T}_{t|t}})^\dagger y_t, \ (\hat{S}_t)_{\hat{T}_{t|t}^c} = 0 \quad (19)$$
- 4) Update centroid by (24), (25) and (26).

In practice, one can replace (21) by $\min_s \|s_{T^c}\|_1$ subject to $\|(I - \hat{P}_t \hat{P}_t')(M_t - s)\|_2 \leq \epsilon$ with $\epsilon = \|(I - \hat{P}_t \hat{P}_t') \hat{L}_{t-1}\|_2$

Predict Support:

Using the location prediction from above, compute the support prediction as

$$\hat{T}_{t|t-1} = [\hat{p}_{t|t-1} - w, \hat{p}_{t|t-1} + w] \quad (20)$$

Recover S_t and Update its Support using Modified-CS:

Assuming $T = \hat{T}_{t|t-1}$ is a good support prediction, we can use it as the partial support knowledge for modCS which solves

$$\min_s \|s_{T^c}\|_1 \text{ subject to } \|y_t - A_t s\|_2 \leq \epsilon \quad (21)$$

i.e., it tries to find the solution that is sparsest outside the set T among all solutions that satisfy that data constraint. Let \hat{s} be the solution of (21) with $T = \hat{T}_{t|t-1}$. As explained in [35], \hat{s} is biased towards zero along T^c (because of the ℓ_1 term) and it may be biased away from zero along T (there is no cost on s_T and the only constraint is the data constraint). The elements which are missing from the support prediction, $\Delta_{t|t-1} := T_t \setminus T$ are a subset of T^c where as the extra elements in the support prediction, $\Delta_{e,t|t-1} := T \setminus T_t$ are a subset of T . If we use a single threshold for support estimation as in (10), we will need a small threshold to ensure that most elements of $\Delta_{t|t-1}$ get correctly detected, but a large one will be needed

Algorithm 3 ReProCS(modCS)

At $t = t_0$, suppose a good estimate of PC matrix, \hat{P}_{t_0} is available from training data. For $t > t_0$, do the following:

- 1) Let $A_t \leftarrow A_{t-1}$. Obtain y_t by (5).
 - 2) Estimate S_t by Algorithm 2.
 - 3) Estimate $\hat{L}_t = M_t - \hat{S}_t$.
 - 4) Update \hat{P}_t using Algorithm 4 which is based on [4].
Update $A_t := (\hat{P}_{t,\perp})'$.
 - 5) Increment t by 1 and go to step 1).
-

to ensure that elements of $\Delta_{e,t|t-1}$ are deleted. Thus for a given threshold, one or the other cannot be done well.

A better approach is to use the Add-LS-Del procedure summarized in step 3 of Algorithm 2. This was first introduced in our older work [19], [36] and simultaneously also in [15], [16]. The addition step threshold, α_{add} , needs to be just large enough to ensure that the matrix used for LS estimation, $A_{T_{\text{add}}}$ is well-conditioned. If α_{add} is chosen properly, the LS estimate on T_{add} will have smaller error than the modCS output. As a result, deletion will be more accurate when done using this estimate. This also means that one can use a larger α_{del} to ensure quicker deletion of extras. We denote the final support estimate output of the Add-LS-Del procedure by $\hat{T}_{t|t}$.

Update Location:

The centroid of the updated support estimate, $\hat{T}_{t|t}$, obtained above, serves as the “observed” location for the KF, $p_{t,\text{obs}}$, i.e.

$$p_{t,\text{obs}} = \text{centroid}(\hat{T}_{t|t}) := \frac{1}{|\hat{T}_{t|t}|} \sum_{i \in \hat{T}_{t|t}} i \quad (22)$$

The following is a valid model for $p_{t,\text{obs}}$

$$p_{t,\text{obs}} = H g_t + \omega_t, \quad H := [1 \ 0] \quad (23)$$

where ω_t is the observation error, which is bounded, zero mean and has a variance R . The observation error arises because there are extras and misses in $\hat{T}_{t|t}$ and hence $p_{t,\text{obs}} = \text{centroid}(\hat{T}_{t|t}) \neq \text{centroid}(T_t) = p_t$. In practice, using the median instead of centroid provides a better observed estimate especially when there are many extras in the support estimate. We discuss how to set R in Sec. IV-D.

The KF update step is as follows [34, Chapter 5].

$$K_t = \Sigma_{t|t-1} H' (H \Sigma_{t|t-1} H' + R)^{-1} \quad (24)$$

$$\hat{g}_{t|t} = \hat{g}_{t|t-1} + K_t (p_{t,\text{obs}} - H \hat{g}_{t|t-1}) \quad (25)$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - K_t H \Sigma_{t|t-1} \quad (26)$$

The above algorithm is summarized in Algorithm 2 and also in the block diagram of Fig.2(b). A complete algorithm incorporating the idea of supp-pred-modCS into ReProCS is given in Algorithm 3.

C. Stability: Main Ideas**

The ReProCS algorithm is a recursive approach. In ReProCS(modCS), the sparse recovery algorithm, support-predicted modified-CS, is itself recursive. Hence an important

question is when and why will it be stable (error bounded by a time-invariant and small value)? Our simulations given in Sec. VI do indicate that it is stable. In this section, we intended to provide the main arguments of why this can be shown analytically. Due to lack of space, we have moved this discussion to Supplementary Material. Based on reviewer comments, it can later be incorporated into the main paper by shortening other sections.

D. Discussion and Extensions**

It is easy to see that, if p_t is the centroid of $\hat{T}_{t|t}$, then the observation error can be bounded as $|\omega_t| \leq |\Delta_{t|t}| \frac{w}{2w+1-|\Delta_{t|t}|} + |\Delta_{e,t|t}| \frac{\max_{j \in \Delta_{e,t|t}} |j-p_t|}{2w}$. We show this in the stability discussion (see Supplementary Material). Denote this bound by B . A possible way to set the variance, R , of the observation error ω_t is to assume that it has the maximum variance distribution for a given bound (i.e. is uniformly distributed). Using this, $R = B^2/3$. But doing this will require roughly knowing the final number of misses and extras. This will depend on the intensity distribution of the moving object and of the background. It can be estimated only if a training sequence with the same object is available. But even without this, the following qualitative fact always holds. If the background and foreground values are quite different, the updated support estimate will be more accurate. So, clearly ω_t will be small. This can also be seen from the bound on $|\omega_t|$ which is directly proportional to the updated support errors. In this case, R should be smaller. In general, R should be of the same order as Q if we want the KF to compute a weighted average of the new observed location and the predicted one. The smaller (larger) the value of R compared to Q , the greater (lesser) the KF depends on the observed location (centroid/median of the updated support).

For the sake of simplicity, we presented our idea of supp-pred-modCS for a 1D image sequence using a very simple correlation model on the support change of the sparse part S_t . The sparse part was assumed to contain a single translating object. However, we can extend it to other more general cases. The extension to 2D image sequences is easy. Position and velocity will each be a two dimensional vector. The support prediction and the location update steps will need simple changes to handle 2D motion. The extension to multiple objects is also easy if they have sufficiently different intensities; if their supports do not overlap for too long; and if their motion is independent. One can have a separate KF for tracking the motion of each object. To obtain the observed locations of the different objects one can use intensity thresholds to obtain their respective supports. This is done in our experiment of Sec. VI-C.

The case of multiple and changing number of objects; objects that can enter and leave the scene; and objects that cannot be separated by intensity thresholding but need more sophisticated segmentation techniques will be studied in future work. Also, in the current work we only assume translating objects. The extension to any other linear model (e.g. affine deformation model in case of video or any other linear model in other applications) is easy. Finally, so far we have not used

the fact that the support will consist of one or a few contiguous blocks in the support update step. Using this can significantly help remove arbitrary extras. One way to use this fact is to use the model-based CS idea of [18].

V. RECURSIVE PCA**

As explained in Sec. I-A, $L_t = Ux_t$ where U is an unknown orthonormal matrix and x_t is a sparse vector. The support set of x_t , N_t , changes every-so-often and the nonzero elements of x_t are spatially uncorrelated. Thus, P_t also changes every-so-often. One realistic generative model on x_t and hence on L_t is given in the Appendix. When new PCs appear, we need to detect them timely before the projected noise, β_t , seen by CS gets too large. When some old PCs vanish, we also need to remove them from \hat{P}_t . Otherwise, $r = \text{rank}(\hat{P}_t)$ will keep increasing and therefore the number of projected measurements, $n - r$, will keep decreasing and may become too small for CS or modCS to work.

At the initial time, $t = t_0$, if the training sequence, $[M_1, \dots, M_{t_0}]$, does not contain any sparse part, we let $[\hat{L}_1, \dots, \hat{L}_{t_0}] = [M_1, \dots, M_{t_0}]$. Usually such a sequence is easy to obtain, e.g. in video this means having a sequence with no foreground moving objects. If $[M_1, \dots, M_{t_0}]$ does contain the sparse part, but its support is small enough or uncorrelated enough so that PCP works, then $[\hat{L}_1, \dots, \hat{L}_{t_0}]$ can be obtained by solving PCP, i.e. (4), with $M = [M_1, \dots, M_{t_0}]$.

The initial PC matrix, \hat{P}_{t_0} , is estimated by computing a singular value decomposition (SVD) of $[\hat{L}_1, \dots, \hat{L}_{t_0}]$, and retaining singular values above a threshold α_0 , i.e., we compute $[\hat{L}_1, \dots, \hat{L}_{t_0}] \stackrel{SVD}{=} P\Lambda V'$; set $T \leftarrow \{i : (\Lambda)_{i,i} > \alpha_0\}$ and set $\hat{P}_{t_0} \leftarrow P_T$, $\hat{\Lambda}_{t_0} \leftarrow \Lambda_{T,T}$, $\hat{V}_{t_0} \leftarrow V_T$. Thus, all but the singular values above α_0 are zeroed to give a truncated SVD that approximates the data. Usually, α_0 is picked according to the distribution of the singular values. If the matrix $[\hat{L}_1, \dots, \hat{L}_{t_0}]$ is exactly low rank, α_0 can be zero. Otherwise it can be picked so that T is the $p\%$ -energy set of $\text{diag}(\Lambda)$ for p large enough.

We use \hat{V}_t to only explain how we update \hat{P}_t , but we do not need to compute and store \hat{V}_t .

For $t > t_0$, we obtain \hat{L}_t by ReProCS. In the recursive PCA step of ReProCS, we update \hat{P}_t every τ frames. Alternatively, we can also do this whenever $\epsilon = \|(I - \hat{P}_{t-1}\hat{P}_{t-1}')\hat{L}_{t-1}\|_2$ exceeds a threshold.

The PC update is done as follows. The complete algorithm is summarized in Algorithm 4. At $t = t_0 + k\tau$, let $D = [\hat{L}_{t-\tau+1}, \dots, \hat{L}_t]$. In step 2a), we first estimate the variance of D along the columns of \hat{P}_{t-1} and remove the PCs along which the variance is below a threshold α . Let σ_{\min} be the smallest nonzero singular value of $\hat{\Lambda}_{t_0}$. We use $\alpha = 0.5\sigma_{\min}^2$. Note that after doing step 2a), the column vectors of \hat{P}_{t-1} contain all the non-decayed PCs.

In step 2b), we rotate \hat{P}_{t-1} and find the new PCs based on the idea of incremental SVD [4]. We first decompose the new collected data D into two components C and E , where $C = \hat{P}_{t-1}'D$ and $E = D - \hat{P}_{t-1}C$. The parallel component C rotates the existing singular vectors and the orthogonal component E estimates the new PCs [4]. Let $E \stackrel{QR}{=} JK$ be a QR decomposition of E . Notice that \hat{P}_{t-1} and J are

Algorithm 4 Recursive PCA

Initialization: Compute $[\hat{L}_1, \dots, \hat{L}_{t_0}] \stackrel{SVD}{=} P\Lambda V'$; set $T \leftarrow \{i : (\Lambda)_{i,i} > \alpha_0\}$, $\hat{P}_{t_0} \leftarrow P_T$, $\hat{\Lambda}_{t_0} \leftarrow \Lambda_{T,T}$. Let $D = \emptyset$. At each time $t > t_0$, store \hat{L}_t in D , i.e., $D \leftarrow [D \ \hat{L}_t]$. Do the following:

- 1) If there are less than τ frames in D ,
 - Let $\hat{P}_t \leftarrow \hat{P}_{t-1}$, $\hat{\Lambda}_t \leftarrow \hat{\Lambda}_{t-1}$.
- 2) If there are τ frames in D
 - 2a) Remove decayed directions from \hat{P}_{t-1}

$$\begin{aligned} \hat{\Sigma}_E &= \frac{1}{\tau} \hat{P}_{t-1}' D D' \hat{P}_{t-1} \\ \hat{N}_E &= \{i : (\hat{\Sigma}_E)_{i,i} \geq \alpha\} \\ \hat{P}_{t-1} &\leftarrow (\hat{P}_{t-1})_{\hat{N}_E}, \quad \hat{\Lambda}_{t-1} \leftarrow (\hat{\Lambda}_{t-1})_{\hat{N}_E, \hat{N}_E} \\ \hat{r}_{t-1} &\leftarrow \text{rank}(\hat{P}_{t-1}) \end{aligned}$$

- 2b) Update \hat{P}_t and $\hat{\Lambda}_t$ by incremental SVD [4]

- Compute $C = \hat{P}_{t-1}'D$ and $E = D - \hat{P}_{t-1}C$
- Compute QR decomposition of E : $E \stackrel{QR}{=} JK$
- Compute SVD of $\begin{bmatrix} \hat{\Lambda}_{t-1} C \\ 0 \quad K \end{bmatrix} : \begin{bmatrix} \hat{\Lambda}_{t-1} C \\ 0 \quad K \end{bmatrix} \stackrel{SVD}{=} P_r \Lambda_r V_r'$
- Update \hat{P}_t and $\hat{\Lambda}_t$ as

$$\hat{P}_t \leftarrow [\hat{P}_{t-1} \ J] P_r, \quad \hat{\Lambda}_t \leftarrow \Lambda_r$$

- 2c) Add new PCs with large variance

$$\begin{aligned} \hat{N}_t &= \{1, \dots, \hat{r}_{t-1}\} \cup \{i > \hat{r}_{t-1} : \frac{(\hat{\Lambda}_t)_{i,i}^2}{\tau} \geq \alpha\} \\ \hat{P}_t &\leftarrow (\hat{P}_t)_{\hat{N}_t}, \quad \hat{\Lambda}_t \leftarrow (\hat{\Lambda}_t)_{\hat{N}_t} \end{aligned}$$

- 2e) Reset $D = \emptyset$.
-

orthogonal, i.e., $\hat{P}_{t-1}'J = 0$. The column vectors of J are the basis vectors of the subspace spanned by the new PCs. It is easy to see that [4]

$$[\hat{P}_{t-1} \hat{\Lambda}_{t-1} \hat{V}_{t-1}' \ D] = [\hat{P}_{t-1} \ J] \begin{bmatrix} \hat{\Lambda}_{t-1} C \\ 0 \quad K \end{bmatrix} \begin{bmatrix} \hat{V}_{t-1} 0 \\ 0 \quad I \end{bmatrix}'$$

Let $\begin{bmatrix} \hat{\Lambda}_{t-1} C \\ 0 \quad K \end{bmatrix} \stackrel{SVD}{=} P_r \Lambda_r V_r'$. Clearly,

$$[\hat{P}_{t-1} \hat{\Lambda}_{t-1} \hat{V}_{t-1}' \ D] \stackrel{SVD}{=} ([\hat{P}_{t-1} \ J] P_r) \Lambda_r \left(\begin{bmatrix} \hat{V}_{t-1} 0 \\ 0 \quad I \end{bmatrix} V_r \right)'$$

In words, P_r rotates the old PCs, \hat{P}_{t-1} , and the new basis vectors, J , to the current PCs, i.e., $\hat{P}_t = [\hat{P}_{t-1} \ J] P_r$. Also, the singular values along \hat{P}_t are the diagonal elements of $\hat{\Lambda}_t = \Lambda_r$.

Let $\hat{r}_{t-1} = \text{rank}(\hat{P}_{t-1})$ and let $\hat{N}_A = \{\hat{r}_{t-1} + 1, \dots, \text{rank}(\hat{P}_t)\}$. If $\hat{P}_{t-1} \approx P_{t-1}$, the old PCs are already correctly estimated and do not need to be rotated. Under this assumption, $(\hat{P}_t)_{\hat{N}_A}$ contains the new PCs. The variance of D along the columns of $(\hat{P}_t)_{\hat{N}_A}$ is given by the diagonal elements of $\frac{1}{\tau} (\hat{\Lambda}_t)_{\hat{N}_A}^2$. Therefore, to only retain the new PCs with large variance, we threshold on $\frac{1}{\tau} (\hat{\Lambda}_t)_{\hat{N}_A}^2$ in step 2c).

Recall that $n - r$ (where $r = \text{rank}(\hat{P}_t)$) is the number of projected measurements and β_t is the noise seen by CS or modified-CS (see (6)). By choosing different values of the threshold, α , there is a tradeoff between making β_t small and keeping $n - r$ large. A smaller α means we retain more directions in \hat{P}_t . This means that $n - r$ is smaller (fewer measurements) but the projected noise, $\beta_t = (I - \hat{P}_t \hat{P}_t') L_t$ is also smaller. When the nonzero elements of the sparse part, S_t , have very small magnitudes, we need a smaller α to ensure that the noise, β_t , is sufficiently small compared to S_t .

VI. EXPERIMENTAL RESULTS

We compare the performance of ReProCS and ReProCS(modCS)¹ with two recursive robust PCA methods – incremental robust subspace learning (iRSL) [10], adapted (outlier-detection enabled) incremental SVD (adapted-iSVD) [4], and two batch robust PCA methods – principal components' pursuit (PCP)² [7], robust subspace learning (RSL)³ [6]. We also show a comparison with simple thresholding based background subtraction (BS) for the last experiment.

In Sec. VI-A and Table I, we compare ReProCS with iRSL[10], adapted-iSVD[4], PCP [7] and RSL [6]. As can be seen, iRSL and adapted-iSVD fail in case of both small magnitude S_t 's and large support-sized S_t 's. RSL can handle large support size, but not small magnitude S_t 's while the opposite is true for PCP. In later comparisons we only compare against PCP and RSL. In Sec. VI-B, Table II and Fig. 3, we give a detailed comparison of ReProCS and PCP. PCP has large error whenever the support sets of the S_t 's are correlated and their sizes are large. Comparison with RSL is also shown.

In Sec. VI-C and Fig. 4, we show an example where ReProCS does not work and ReProCS(modCS) is needed. Here the support size of S_t , $|T_t| \approx 0.51n$ while the number of projected measurements is only about $n - r \approx 0.8n$.

In Sec. VI-D, we show comparisons on a partly-real video sequence where we overlay simulated sparse foreground images on real background images. The background sequence has nonzero mean. The background variations, e.g. those due to water waves, lead to the low-rank part having covariance matrix rank as large as 20% of image size. Because the variations due to the water waves are not localized in one or more small image regions, the resulting projection matrix, $A_t = (\hat{P}_{t,\perp})'$, is “incoherent” enough compared to the sparse foreground image and so CS and modCS work.

All our code will be posted on the first author's webpage, <http://www.ece.iastate.edu/~chenlu>. Videos of all the above experiments and some results on fully real video sequences are also posted here.

¹we use YALL1 ℓ_1 minimization toolbox [37] to solve (7) and (21). Its code is available at <http://yall1.blogs.rice.edu/>.

²We use Accelerated Proximal Gradient algorithm[38] and Inexact ALM algorithm [39] (designed for large scale problems) to solve PCP (4). The code is available at http://perception.csl.uiuc.edu/matrix-rank/sample_code.html.

³The code of RSL is available at <http://www.salleurl.edu/~ftorre/papers/rpca/rpca.zip>.

A. Comparison of ReProCS with adapted-iSVD, iRSL, RSL and PCP

In this experiment, the measurement at time t , $M_t := L_t + S_t$, is an $n \times 1$ vector with $n = 100$. The low rank part, $L_t = Ux_t$, where U is an $n \times n$ orthonormal matrix (generated by first generating an $n \times n$ matrix with entries randomly sampled from a standard Gaussian distribution, and then orthonormalizing it using the Gram-Schmidt process) and x_t is an $n \times 1$ sparse vector with support set N_t with size $|N_t| \approx 0.2n$. Thus the PC matrix at time t is $P_t := (U)_{N_t}$. The sparse vector, x_t , is simulated using the regression model described in the Appendix with $f = 0.5$, $f_d = 0.1$, and $\theta = 0.5$. Initially there are 20 PCs with variances $10^4, 0.7079 \times 10^4, 0.7079^2 \times 10^4, \dots, 14.13$. At $t = t_0 + 5$, two new PCs with variance 50 and 60 enter the PC basis, P_t , and the values of x_t along two old PCs start to decay exponential to zero. The sparse part, S_t , is an $n \times 1$ vector with its support set denoted by T_t . For $1 \leq t \leq t_0 = 2000$, $S_t = 0$ and hence $M_t = L_t$. For $t > t_0$, the support set of S_t , T_t , is generated in a correlated fashion: S_t contains either one or four nonzero strips (small and large support size cases). Each strip has 9 nonzero elements and it can either stay static with a large probability 0.8 or move top/down with a small probability 0.1 independently of the others. Thus the support sets are both spatially and temporally highly correlated. The magnitude of the nonzero elements of S_t is fixed at either 100 (large) or 10 (small). The two cases are plotted in Fig. 1.

We tabulate our results in Table I. The first t_0 frames can be used to obtain the initial PC matrix, \hat{P}_{t_0} , by standard PCA on $[M_1, \dots, M_{t_0}] = [L_1, \dots, L_{t_0}]$. The same \hat{P}_{t_0} is used by ReProCS (Algorithm 1) and by the two other recursive algorithms – iRSL [10] and adapted-iSVD [4]. For ReProCS, the support estimation threshold in (10), γ , can be set a little lower than the minimum nonzero magnitude that we would like to correctly detect. Thus we set $\gamma = a \min_{i \in T_t} |(S_t)_i|$ for an $a < 1$. We used $a = 0.2$ in the large magnitude S_t case, but $a = 0.3$ in the small magnitude S_t case (using a larger a in this case ensures that γ is greater than the noise level). We update \hat{P}_t for every $\tau = 20$ frames by Algorithm 4 with $\alpha = 5$. iRSL [10] solves the recursive robust PCA problem by weighting each data point according to its reliability (thus soft-detecting and down-weighting likely outliers). If the final goal is outlier detection (e.g. detecting foreground moving objects), this is done in [10] by thresholding on the difference between the current data vector and its projection into the PCA space, i.e. by thresholding on $(I - \hat{P}_t \hat{P}_t') M_t$. We also compare against an adapted version of [4] (we call it adapted-iSVD). We provide adapted-iSVD the outlier locations also by thresholding on $(I - \hat{P}_t \hat{P}_t') M_t$. It fills in the corrupted locations of L_t by imposing that L_t lies in $\text{span}(\hat{P}_t)$. We used a threshold of $0.5 \min_{i \in T_t} |(S_t)_i|$ for both iRSL and adapted-iSVD (we also tried various other options for thresholds but with similar results). PCP [7] is a batch method that needs to wait until all data frames $M = [M_1, \dots, M_t]$ are available and then estimates the low rank part $L = [L_1, \dots, L_t]$ and the sparse part $S = [S_1, \dots, S_t]$ simultaneously by solving the convex optimization problem of (4) with $\lambda = 1/\sqrt{\max(n, t)}$.

as suggested in [7]. RSL [6] is another state-of-art batch robust PCA method that aims at recovering a good low-rank approximation that best fits the majority of the data. RSL solves a nonconvex optimization via alternative minimization based on the idea of soft-detecting and down-weighting the outliers. For RSL and PCP, \hat{S}_t is not exactly sparse and therefore we estimate the support, \hat{T}_t , by thresholding on \hat{S}_t with a threshold picked in the same way for adapted-iSVD and iRSL.

iRSL was designed for a video application and hence it directly detects the foreground overlay's support by thresholding on $(I - \hat{P}_t \hat{P}_t') M_t$ (it never computes S_t). Thus, in this experiment, for all algorithms, we compare the reconstruction error of the foreground overlay, $(O_t)_{T_t} = (M_t)_{T_t}$ and $(O_t)_{T_t^c} = 0$, where T_t is the support set of S_t . In Table I, we compare the normalized mean square error (MSE), $\mathbb{E}\|O - \hat{O}\|_F^2 / \mathbb{E}\|O\|_F^2$ where $O := [O_{t_0+1}, \dots, O_{t_0+100}]$. The expected value is computed using 100 times Monte Carlo averaging.

As can be seen, ReProCS gives small error in all four cases. ReProCS is able to successfully recover both small magnitude and fairly large support-sized S_t 's because it operates by first approximately nullifying L_t , i.e. computing $y_t := (I - \hat{P}_t \hat{P}_t') M_t$, and then recovering S_t by solving (7) that enforces the sparsity of S_t . Adapted-iSVD and iRSL also approximately nullify L_t by computing the same y_t , but they directly use y_t to detect or soft-detect (and down-weight) the support of S_t . Using (1), y_t can be rewritten as $y_t = S_t + (-\hat{P}_t \hat{P}_t' S_t) + \beta_t$ where $\beta_t := (I - \hat{P}_t \hat{P}_t') L_t$. As the support of S_t increases, the interference due to $(-\hat{P}_t \hat{P}_t' S_t)$ becomes large, resulting in wrong estimates of S_t . Since adapted-iSVD and iRSL are recursive methods, this, in turn, results in wrong PC matrix updates, thus also causing β_t to become large and finally causing the error to blow up. Using y_t directly to detect the support of S_t also becomes difficult when the magnitude of the nonzero S_t 's is small compared to β_t and this is another situation where adapted-iSVD and iRSL fail. RSL is able to handle larger support size of S_t 's because it uses the entire sequence of $t_0 + 100$ M_t 's jointly and the first $t_0 = 2000$ M_t 's have $S_t = 0$. But when the magnitude of the nonzero S_t 's is small, RSL also fails. PCP is able to always deal with small support sized S_t 's (even when their magnitude is small). But it fails for large support sizes of S_t 's, particularly since they are also correlated. PCP is much more robust to independently generated S_t 's with similar ratios of $|T_t|/(n - r)$ (shown next).

B. Recovery of random or structured sparse outliers

In this experiment, we consider the problem of tracking moving blocks in a simulated image sequence of size $32 \times 32 \times (t_0 + 200)$. Thus, the data vector at time t , M_t , is $n = 32^2 = 1024$ dimensional. Note that $M_t = L_t + S_t$ with L_t and S_t generated as described below. We let $L_t = U x_t$ where U is an orthonormal $n \times n$ matrix generated as before and x_t is an n dimensional sparse vector. The sparse vector, x_t , is simulated using the regression model described in the Appendix with $f = 0.5$, $f_d = 0.1$, and $\theta = 0.5$. In Table III(a) and Table III(b), initially there are 32 PCs with variances

$10^4, 0.8058 \times 10^4, 0.8058^2 \times 10^4, \dots, 12.4$. In Table III(c) and Table III(d), initially there are 128 PCs with variances $10^4, 0.9471 \times 10^4, 0.9471^2 \times 10^4, \dots, 10$. In all four cases, at time $t = t_0 + 5$, two new PCs with variance 50 and 60 enter the PCs basis and the value of x_t along two old PCs begins to decay exponentially to zero. At time $t = t_0 + 50$, another two new PCs with variance 55 and 65 enter the PCs basis and the value of x_t along two old PCs begins to decay exponentially to zero. For $1 \leq t \leq t_0$, $S_t = 0$ and hence $M_t = L_t$. For $t > t_0$, S_t has nonzero elements. Each nonzero element of S_t has constant value 5, which is much smaller than those of L_t 's. We generate the support of S_t , T_t , in the following two different ways:

- In Table III(a) and Table III(c), for a given support size $|T_t|$, T_t is generated uniformly at random.
- In Table III(b) and Table III(d), T_t is spatially and temporal correlated as described below. At each time $t > t_0$, there are several 7×7 blocks having constant pixel value 5. All other entries in S_t are zero. Each block can either stay static with probability 0.8 or move independently one pixel step towards top/bottom/left/right with probability 0.05.

For ReProCS, we use the first $t_0 = 10^4$ frames, $[M_1, \dots, M_{t_0}] = [L_1, \dots, L_{t_0}]$, as training data to get an initial PCs' estimate \hat{P}_{t_0} via computing its SVD. For $t > t_0$, we do Algorithm 1 to recursively estimate L_t and S_t . The support estimation threshold in (10), γ , can be set a little lower than the minimum nonzero magnitude that we would like to correctly detect. We set $\gamma = 0.2 \min_{i \in T_t} |(S_t)_i| = 1$. We update \hat{P}_t for every $\tau = 20$ frames by Algorithm 4 with $\alpha = 5$. The reconstruction time of ReProCS is about 0.015 seconds per frame. To make a fair comparison, we use the entire sequence, $[M_1, \dots, M_{t_0+200}]$, to do PCP[7] and RSL[6]. The reconstruction time of offline PCP, (total time)/($t_0 + 200$), is about 0.05 seconds per frame⁴. The reconstruction time of offline RSL, (total time)/($t_0 + 200$), is about 0.03 seconds per frame. But if we had to do PCP or RSL for a causal reconstruction, i.e., at each time t , estimate S_t and L_t by solving PCP or RSL using all available data $[M_1, \dots, M_t]$, PCP will take about 500 seconds and RSL will take about 300 seconds at each time.

We summarize the reconstruction errors of PCP and ReProCS in Table II based on 100 times Monte Carlo averaging. The normalized MSE $\mathbb{E}(\|L - \hat{L}\|_F^2) / \mathbb{E}(\|S\|_F^2)$ and $\mathbb{E}(\|S - \hat{S}\|_F^2) / \mathbb{E}(\|S\|_F^2)$ are computed only for the last 200 frames. As can be seen from Table II, PCP has large reconstruction error for less sparse and/or correlated S_t 's. However, ReProCS can recursively recover L_t and S_t accurately, with reconstruction error less than 10^{-2} in all cases. In Fig. 3(a) and Fig. 3(b), we plot the normalized MSE (NMSE) of L_t and S_t for the third case in Table III(d) where $\frac{|T_t|}{n} \approx 28.71\%$. We also show the error of RSL. Since the nonzero elements of S_t have much smaller magnitude than the elements of L_t , RSL also does not work. In Fig. 3(c), we show \hat{S}_{t_0+200} obtained by ReProCS, PCP, RSL [6] as an image.

⁴PCP using the last 200 frames takes about 0.02 seconds per frame but it gives even larger errors for L_t and S_t

(a) $(S_t)_i = 100, \forall i \in T_t$ and $(S_t)_i = 0, \forall i \in T_t^c$

$ T_t /n$	ReProCS	adapted-iSVD	iRSL	RSL	PCP
9%	0.00002	0.0381	0.9817	0.0001	0
36%	0.0119	0.3172	0.9813	0.0445	0.5553

(b) $(S_t)_i = 10, \forall i \in T_t$ and $(S_t)_i = 0, \forall i \in T_t^c$

$ T_t /n$	ReProCS	adapted-iSVD	iRSL	RSL	PCP
9%	0.0480	0.1949	0.9890	4.9618	0.0946
36%	0.0563	0.4043	0.9920	1.2482	0.5786

TABLE I: Normalized MSE $\mathbb{E}\|O - \hat{O}\|_F^2 / \mathbb{E}\|O\|_F^2$ of various methods. The experiment is described in Sec. VI-A. Here, $|T_t|/n$ is the sparsity ratio of S_t .(a) $\text{rank}(L) = 36$, $\text{supp}(S)$ is uniformly random

$ T_t /n$	PCP		ReProCS	
	$\frac{\mathbb{E}\ L - \hat{L}\ _F^2}{\mathbb{E}\ L\ _F^2}$	$\frac{\mathbb{E}\ S - \hat{S}\ _F^2}{\mathbb{E}\ S\ _F^2}$	$\frac{\mathbb{E}\ L - \hat{L}\ _F^2}{\mathbb{E}\ L\ _F^2}$	$\frac{\mathbb{E}\ S - \hat{S}\ _F^2}{\mathbb{E}\ S\ _F^2}$
9.57%	1.42×10^{-4}	3.0×10^{-3}	1.32×10^{-4}	2.8×10^{-3}
19.14%	5.58×10^{-4}	5.8×10^{-3}	1.81×10^{-4}	1.9×10^{-3}
28.71%	1.80×10^{-3}	1.25×10^{-2}	2.42×10^{-4}	1.7×10^{-3}

(b) $\text{rank}(L) = 36$, $\text{supp}(S)$ is correlated

$ T_t /n$	PCP		ReProCS	
	$\frac{\mathbb{E}\ L - \hat{L}\ _F^2}{\mathbb{E}\ L\ _F^2}$	$\frac{\mathbb{E}\ S - \hat{S}\ _F^2}{\mathbb{E}\ S\ _F^2}$	$\frac{\mathbb{E}\ L - \hat{L}\ _F^2}{\mathbb{E}\ L\ _F^2}$	$\frac{\mathbb{E}\ S - \hat{S}\ _F^2}{\mathbb{E}\ S\ _F^2}$
9.57%	9.6×10^{-4}	0.022	2.80×10^{-4}	6.4×10^{-3}
19.14%	2.76×10^{-2}	0.333	4.75×10^{-4}	5.7×10^{-3}
28.71%	5.81×10^{-2}	0.502	6.52×10^{-4}	5.6×10^{-3}

(c) $\text{rank}(L) = 132$, $\text{supp}(S)$ is uniformly random

$ T_t /n$	PCP		ReProCS	
	$\frac{\mathbb{E}\ L - \hat{L}\ _F^2}{\mathbb{E}\ L\ _F^2}$	$\frac{\mathbb{E}\ S - \hat{S}\ _F^2}{\mathbb{E}\ S\ _F^2}$	$\frac{\mathbb{E}\ L - \hat{L}\ _F^2}{\mathbb{E}\ L\ _F^2}$	$\frac{\mathbb{E}\ S - \hat{S}\ _F^2}{\mathbb{E}\ S\ _F^2}$
9.57%	1.69×10^{-4}	1.32×10^{-2}	4.98×10^{-5}	3.9×10^{-3}
19.14%	6.55×10^{-4}	2.55×10^{-2}	6.89×10^{-5}	2.7×10^{-3}
28.71%	1.70×10^{-3}	4.33×10^{-2}	1.05×10^{-4}	2.7×10^{-3}

(d) $\text{rank}(L) = 132$, $\text{supp}(S)$ is correlated

$ T_t /n$	PCP		ReProCS	
	$\frac{\mathbb{E}\ L - \hat{L}\ _F^2}{\mathbb{E}\ L\ _F^2}$	$\frac{\mathbb{E}\ S - \hat{S}\ _F^2}{\mathbb{E}\ S\ _F^2}$	$\frac{\mathbb{E}\ L - \hat{L}\ _F^2}{\mathbb{E}\ L\ _F^2}$	$\frac{\mathbb{E}\ S - \hat{S}\ _F^2}{\mathbb{E}\ S\ _F^2}$
9.57%	4.61×10^{-4}	0.04	1.03×10^{-4}	9.1×10^{-3}
19.14%	7.5×10^{-3}	0.35	1.74×10^{-4}	8.1×10^{-3}
28.71%	1.64×10^{-2}	0.53	2.47×10^{-4}	8.1×10^{-3}

TABLE II: Comparison of reconstruction errors of PCP[7] and ReProCS for separating low rank L and sparse S . Here, L and S are of size $n \times 200$ and $n = 1024$; the rank of L , $\text{rank}(L)$, is much smaller than n ; and the support of S is generated either uniformly at random or correlated. Note that $|T_t|$ is the number of nonzero elements in S_t .

We also verified that ReProCS keeps tracking the change of PCs gradually and correctly. At $t = t_0$, we have a good \hat{P}_{t_0} that finds all the existing PCs. At $t = t_0 + 5$, let U_1 denote the two new PCs that get added and let U_2 denote the two old PCs that start to be removed from $\text{span}(\hat{P}_t)$. At $t = t_0$, we have $\frac{\|\hat{P}_t' U_1\|_F^2}{\|U_1\|_F^2} = 0$ (U_1 is not in $\text{span}(\hat{P}_t)$) and $\frac{\|\hat{P}_t' U_2\|_F^2}{\|U_2\|_F^2} = 1$ (U_2 is in $\text{span}(\hat{P}_t)$). We updated \hat{P}_t for every $\tau = 20$ frames using Algorithm 4 with $\alpha = 0.5\sigma_{\min}^2 = 5$. At $t = t_0 + 200$, $\frac{\|\hat{P}_t' U_1\|_F^2}{\|U_1\|_F^2}$ increased from 0 to 0.91 and $\frac{\|\hat{P}_t' U_2\|_F^2}{\|U_2\|_F^2}$ decreased from 1 to 0.11. Therefore, new PCs gradually and correctly got added to $\text{span}(\hat{P}_t)$ and the decayed PCs gradually got removed from $\text{span}(\hat{P}_t)$. As explained in Sec. V, we can remove the decayed PCs more quickly by using a larger threshold α .

C. Using correlation of the sparse part: the need for *supp-pred-modCS* and *ReProCS (modCS)*

In this section, we show an example where ReProCS (modCS) is needed. The image sequence is of size $64 \times 80 \times (t_0 + 100)$ with $t_0 = 5000$. Thus, the measurement at time t , M_t , is of length $n = 64 \times 80 = 5120$. We generate $M_t = L_t + S_t$ as follows. We set $L_t = Ux_t$ where U is generated as before and x_t follows the regression model of the Appendix with $f = 0.5$, $f_d = 0.1$, and $\theta = 0.5$. Initially, there are $0.2n$ PCs with variance $10^4, 0.9933 \times 10^4, 0.9933^3 \times 10^4, \dots, 10$. At $t = t_0 + 5$, two new PCs with variance 50 and 55 get added and the value of x_t along two old PCs starts to decay to zero. Compared to the previous experiment, the rank of the PC matrix, $r = \text{rank}(\hat{P}_t) \approx 0.2n$, is larger. For $1 \leq t \leq t_0$, $S_t = 0$ and hence $M_t = L_t$. For $t > t_0$, there are two 45×29 moving objects in S_t . Thus, the support size of S_t is $|T_t| \approx 0.51n$ while the number of projected measurements, y_t , is $n - r \approx 0.8n$. The two blocks move independently following the motion model (13) where n_t is a zero mean truncated Gaussian noise with variance $Q = 2.5 \times 10^{-5}$,

i.e., $n_t \sim \mathcal{N}(0, Q)$ and $-2\sqrt{Q} < |n_t| < 2\sqrt{Q}$. The first block has constant pixel intensity 10 and it moves from left to right with initial velocity $v_{t_0+1} = 0.25$. The second block has constant intensity 20 and it moves from right to left with $v_{t_0+1} = -0.25$. We use a small Q because we want the blocks to stay in the scene for a long sequence (in the next section, we use a larger Q).

For ReProCS(modCS), at time $t = t_0 + 1$, we start a separate KF for tracking the motion of each object with its true location and velocity, i.e. we use $\hat{g}_{t_0+1|t_0} = [p_{t_0+1}, v_{t_0+1}]$, $\Sigma_{t_0+1|t_0} = 0$ for each object. The deletion threshold, α_{del} , can be set a little lower than the minimum nonzero magnitude that we would like to correctly detect. Thus we set $\alpha_{\text{del}} = a \min_{i \in N_t} |(S_t)_i| = 10a$ for an $a < 1$. We used $a = 0.1$ and thus set $\alpha_{\text{del}} = 1$. The addition threshold usually can be much lower to ensure most elements get correctly added. This can either be indirectly set (we can keep adding more elements to T_{add} until the condition number of $(A_t)_{T_{\text{add}}}$ goes below a “well-conditioned” threshold) or we can just set it to a percentage of α_{del} . In this work we used this latter approach and set $\alpha_{\text{add}} = 0.5\alpha_{\text{del}}$. To obtain the observed locations of the different objects, we first use intensity thresholds to obtain their respective supports. The observed location of each block is then obtained as the median of its estimated support, which is more robust than (22) if there are occasionally some extra indices far away from the true support set. We use $R = 4Q = 10^{-4}$ as the variance of the observation noise, w_t , in (23). We use a larger R because the intensities of the moving objects are small compared to the background and hence the support update error is likely to be larger. For both ReProCS and ReProCS(modCS), we update \hat{P}_t for every $\tau = 20$ frames with $\alpha = 0.5\sigma_{\min}^2 = 5$. For ReProCS, we use $\gamma = 1$ for support thresholding in (10).

In practice, foreground objects slowly enter the scene and the support size of S_t , $|T_t|$, increases over time (until they

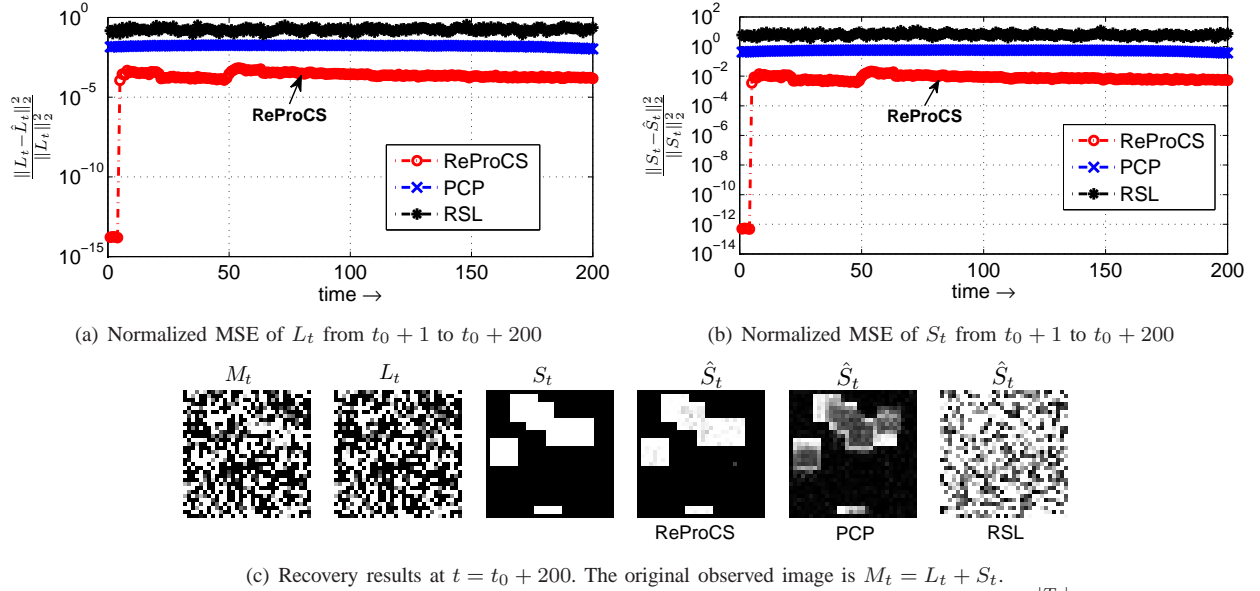


Fig. 3: Normalized MSE plots and reconstruction results for the third case in Table III(d) where $\frac{|T_t|}{n} \approx 28.71\%$

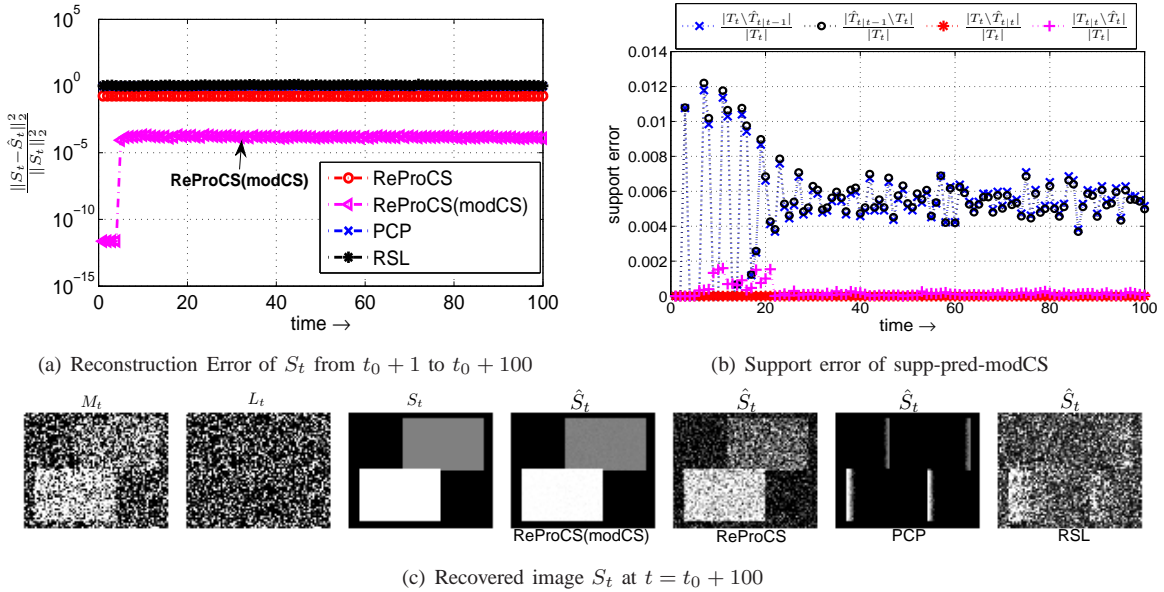


Fig. 4: Comparison of ReProCS and ReProCS(modCS) when limited projected measurements are available. Here, $|T_t| = |\text{supp}(S_t)| \approx 0.51n$ while the number of projected measures is about $n - \text{rank}(P_t) \approx 0.8n$.

enter the scene completely). Therefore, $|T_t|$ shall be small for the initial frames. Even though we may not have the initial location/velocity knowledge, we can do ReProCS for the first a few frames, get initial estimates of the support and then use intensity based segmentation followed by centroid computation to estimate the initial location/velocity of the objects. For the future frames when $|T_t|$ is large, we can replace CS by supp-pred-modCS, i.e., use ReProCS(modCS).

In Fig. 4(a), we plot the NMSE of S_t for ReProCS and ReProCS(modCS) based on 100 times Monte Carlo averaging. We also plot the reconstruction error of PCP and RSL for one realization. ReProCS takes about 2.1 seconds per frame and ReProCS(modCS) takes about 2.8 seconds per frame. As can be seen, with a large $\frac{|T_t|}{n-r}$, ReProCS(modCS) outperforms ReProCS greatly because it utilizes the correlation model of S_t while ReProCS does not. PCP and RSL again do not work. In Fig. 4(b), we plot the average number of extras and misses in

$\hat{T}_{t|t-1}$ (predicted support) and $\hat{T}_{t|t}$ (updated support) used by supp-pred-modCS. Clearly, modCS corrects most prediction errors.

D. ReProCS and ReProCS(modCS) on a real background sequence with simulated foreground images

In this experiment, we compare ReProCS and ReProCS(modCS) with PCP, RSL, and thresholding based background subtraction (BS) on a partly-real video sequence. We take a $72 \times 90 \times 1500$ real video sequence around a lake with slow global background variations, e.g., water waves. By arranging each image frame as a $n = 72 \times 90 = 6480$ dimensional column vector, we get the background sequence, $[L_1, \dots, L_{1500}]$, which has nonzero mean. The first $t_0 = 1420$ frames serve as the training data. The nonzero background mean, denoted by μ_0 , is estimated as the empirical mean of the training sequence, i.e., $\mu_0 = (L_1 + \dots + L_{t_0})/t_0$. We

subtract μ_0 from L_t and then get the initial PCs estimate, \hat{P}_{t_0} , by computing a SVD decomposition of $[L_1 - \mu_0, \dots, L_{t_0} - \mu_0]$. We get $r = \text{rank}(\hat{P}_{t_0}) = 1255 \approx 0.2n$. For $1 \leq t \leq t_0$, we let $M_t = L_t$. For $t > t_0$, we *overlay* a simulated foreground image, O_t , on the background image L_t . The observed image, M_t , is thus $(M_t)_{T_t} = (O_t)_{T_t}$ and $(M_t)_{T_t^c} = (L_t)_{T_t^c}$. Thus, S_t follows (2). The foreground image has one 45×25 moving block with nonzero pixel intensity 200. The foreground block moves slowly from left to right following the motion model (13) with initial velocity $v_{t_0+1} = 0.5$ and velocity change variance $Q = 0.005$ (n_t is truncated Gaussian noise with $n_t \sim \mathcal{N}(0, Q)$ and $-2\sqrt{Q} < |n_t| < 2\sqrt{Q}$). We show M_t , L_t , O_t , and S_t (shown in 2D fashion) at $t = t_0 + 20$ in Fig. 5(c). Compared with the experiments in Sec. VI-B and Sec. VI-C, the magnitudes of most of the nonzero elements of S_t 's are now larger but they vary over location and time. In this case $\min_{i \in T_t} |(S_t)_i| \approx 25$.

Note that L_t 's have nonzero mean μ_0 and \hat{P}_t estimates the PCs of $[L_1 - \mu_0, \dots, L_t - \mu_0]$ instead of $[L_1, \dots, L_t]$. We do ReProCS and ReProCS(modCS) on $M_t - \mu_0$ to estimate \hat{S}_t and then let $\hat{L}_t = M_t - \hat{S}_t$. We do Algorithm 4 to update \hat{P}_t for every 10 frames with $D \leftarrow [D \ \hat{L}_t - \mu_0]$ and $\alpha = 0.1$. For ReProCS, we use $\gamma = 10$ for support thresholding in (10). For ReProCS(modCS), we run a KF in the same way as in Sec. VI-C. We use $R = Q/50 = 10^{-4}$ in (23) since S_t 's now have larger magnitude and so we expect the modCS output to be more reliable. We set $\alpha_{\text{del}} = 0.8 \min_{i \in T_t} |(S_t)_i| = 20$ and $\alpha_{\text{add}} = 0.5\alpha_{\text{del}} = 10$. We do PCP and RSL using the entire sequence, $[M_1, \dots, M_{1500}]$, and get \hat{S}_t . The support estimation, \hat{T}_t , is estimated as the 90%-percent energy set of \hat{S}_t . The foreground estimation is $(\hat{O}_t)_{\hat{T}_t} = (M_t)_{\hat{T}_t}$, $(\hat{O}_t)_{\hat{T}_t^c} = 0$. BS first subtracts the background mean, μ_0 , from the current image observation, M_t , and then estimate O_t by thresholding on the difference image, $M_t - \mu_0$, using a threshold picked in the same way for PCP or RSL.

As can be seen from Fig. 5, the reconstruction errors of PCP, RSL, BS and ReProCS are all much larger than ReProCS(modCS). The recovered image frames at $t = 20$ in shown in Fig. 5(c). In this experiment, the support size of S_t is about $|T_t| \approx 0.16n$ and the number of projected measurements is about $n - r \approx 0.8n$, which seems to be enough for recovering S_t using ReProCS. However, as shown in the figure, ReProCS cannot recover S_t very well. This is because that there is a lot of background variation hence the projection matrix, $A_t = (\hat{P}_{t,\perp})'$, cannot make the noise β_t sufficiently small compared to S_t . When the noise β_t is large, for a given support size $|T_t|$, more measurements are needed for accurate sparse recovery. We did the same experiment but with a smaller foreground block of size 25×19 ($|T_t| \approx 0.07n$). In that case, ReProCS and ReProCS(modCS) give similar reconstruction errors which are much smaller than PCP, RSL, and BS (not shown).

In Fig. 5(c), there are some misses and extras in \hat{O}_t estimated by ReProCS(modCS). This is because the magnitudes of nonzero elements of S_t vary over location and time. Also, as explained earlier, the noise β_t may not be sufficiently small as compared with the small elements of S_t . A possible solution is to use location and time varying thresholds. Also, using

the fact that S_t is block sparse can significantly help improve performance.

VII. CONCLUSIONS, LIMITATIONS AND FUTURE WORK

This work finds the missing link between the recursive robust PCA problem and the recursive sparse recovery problem in large noise. From the robust PCA perspective, our proposed solutions, ReProCS and ReProCS(modCS), are robust to correlated “outliers” as long as they are sparse. From the sparse recovery perspective, they are robust to large “noise” as long as the “noise” is spatially correlated enough to have an approximately low rank covariance matrix and this covariance matrix is either constant or changes slowly with time. Our second solution, ReProCS(modCS), which utilizes the correlation model of the sparse part, can handle significantly less sparse S_t 's or significantly larger ranks of the low rank part.

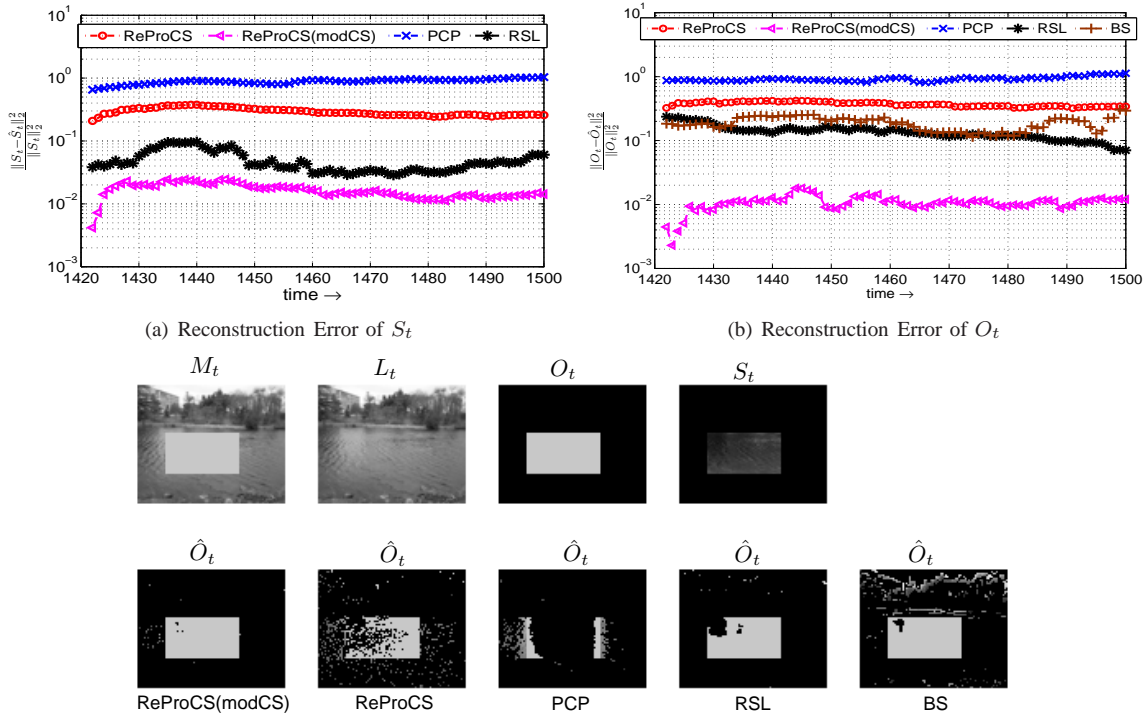
A limitation of our approach, and of recursive robust PCA in general, is that it requires an initial estimate of the PC matrix, \hat{P}_0 . To get that we either need to know that a certain initial set of frames have no sparse part, or we need to use PCP to recover the low rank part and hence estimate \hat{P}_0 . Additionally, it also requires that the L_t 's are spread out enough (not sparse) and that the low dimensional subspace in which L_t lies changes slowly over time. ReProCS(modCS) also needs to know the structure of the correlation model on the sparse part.

In this work we study the case of recursively recovering a sparse vector from full measurements, i.e. $M_t = S_t + L_t$, when the corrupting noise has large magnitude, but is highly spatially correlated (has a low rank covariance matrix). But both ReProCS and ReProCS(modCS) will extend directly to the more general case of $M_t = \Psi S_t + L_t$ where Ψ may be a fat or a square matrix. The only change will be that we will need to use $A_t = (\hat{P}_{t,\perp})'\Psi$ instead of just $A_t = (\hat{P}_{t,\perp})'$ in both Algorithm 1 and Algorithm 3. Also, we will need the assumption that $(\hat{P}_{t,\perp})'\Psi$ does not nullify the sparse vectors S_t .

This work introduces the idea of *support-predicted modified-CS* using one very simple correlation model. Similar ideas can be extended to many more general models and to many applications other than video surveillance. Also, in certain cases, the support change may be slow enough so that the previous support estimate may itself be a very good prediction of the current support. In this case the correlation model update using a KF is not needed. For the video application, certain practical issues are discussed in Sec. IV-D. Moreover, so far we only use support prediction. We can also try to use (a) signal value prediction and (b) incorporate spatial correlation, e.g. using [18]. Finally, the arguments to obtain the conditions under which stability (time-invariant and small bound on the error) of support-predicted-modified-CS holds need to be formalized in future work.

APPENDIX: MODEL ON THE SUPPORT CHANGE OF x_t

We provide here a realistic generative model for x_t and hence the low rank part, $L_t = Ux_t$ that satisfies the assumptions given in Sec. I-A. The support set of x_t , N_t , is a union



(a) Reconstruction Error of S_t (b) Reconstruction Error of O_t

(c) First row: original image at $t = t_0 + 20$. Second row: recovered foreground image

Fig. 5: Here, $\text{rank}(P_t) \approx 0.2n$, and $|T_t| = |\text{supp}(S_t)| \approx 0.16n$.

of three disjoint sets \mathcal{A}_t , \mathcal{D}_t , and \mathcal{E}_t , i.e., $N_t = \mathcal{A}_t \cup \mathcal{D}_t \cup \mathcal{E}_t$. At $t = 1$, we let $N_t = \mathcal{E}_t$ and $\mathcal{A}_t \cup \mathcal{D}_t = \emptyset$. For $t > 1$, the new addition set $\mathcal{A}_t := N_t \setminus N_{t-1}$ is the set of indices along which $(x_t)_{\mathcal{A}_t}$ starts to be nonzero. \mathcal{A}_t is nonempty once every d frames. The set $\mathcal{D}_t \subset (N_t \cap N_{t-1})$ is the set of indices along which $(x_t)_{\mathcal{D}_t}$ decay exponentially to zero. We assume that \mathcal{D}_t will not get added to N_t at any future time. The set $\mathcal{E}_t := N_t \cap N_{t-1} \setminus \mathcal{D}_t$ is the set of indices along with $(x_t)_{\mathcal{E}_t}$ follows a first order autoregressive (AR-1) model.

Let Σ be a diagonal matrix with nonnegative diagonal elements σ_i^2 s. We model x_t as

$$\begin{aligned} x_0 &= 0 \\ x_t &= F_t x_{t-1} + \nu_t, \quad \nu_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, Q_t) \end{aligned} \quad (27)$$

where ν_t is independent and identically distributed Gaussian noise with zero mean and diagonal covariance matrix Q_t ; F_t and Q_t are two diagonal matrices defined as below

$$F_t = \begin{bmatrix} 0_{\mathcal{A}_t} & 0 & 0 & 0 \\ 0 & (fI)_{\mathcal{E}_t} & 0 & 0 \\ 0 & 0 & (f_d I)_{\mathcal{D}_t} & 0 \\ 0 & 0 & 0 & (0)_{N_t^c} \end{bmatrix}$$

$$Q_t = \begin{bmatrix} \theta(\Sigma)_{\mathcal{A}_t} & 0 & 0 & 0 \\ 0 & (1 - f^2)(\Sigma)_{\mathcal{E}_t} & 0 & 0 \\ 0 & 0 & (0)_{\mathcal{D}_t} & 0 \\ 0 & 0 & 0 & (0)_{N_t^c} \end{bmatrix}$$

The three scalars f , f_d , and θ satisfy $0 < f_d < f < 1$ and $0 < \theta < 1$.

From the model on x_t given in (27), we notice the following:

- a) At time $t = jd$, $(x_t)_{\mathcal{A}_t}$ starts with

$$(x_t)_{\mathcal{A}_t} \sim \mathcal{N}(0, \theta(\Sigma)_{\mathcal{A}_t}).$$

Small θ ensures that new directions get added at a small value and increase slowly. $(x_t)_{\mathcal{D}_t}$ decays as

$$(x_t)_{\mathcal{D}_t} = f_d (x_t)_{\mathcal{D}_t}$$

$(x_t)_{\mathcal{E}_t}$ follows an AR-1 model with parameter f :

$$(x_t)_{\mathcal{E}_t} = f(x_{t-1})_{\mathcal{E}_t} + (v_t)_{\mathcal{E}_t}$$

- b) At time $t > jd$, the variance of $(x_t)_{\mathcal{A}_{jd}}$ gradually increases as

$$(x_t)_i \sim \mathcal{N}(0, (1 - (1 - \theta)f^{2(t-jd)})(\Sigma)_{i,i}), \quad i \in \mathcal{A}_{jd}$$

Eventually, the variance of $(x_t)_{\mathcal{A}_{jd}}$ converges to $(\Sigma)_{\mathcal{A}_{jd}}$. We assume this becomes approximately true much before $t = (j+1)d$ (the next support change time).

- c) At time $t > jd$, the variance of $(x_t)_{\mathcal{D}_{jd}}$ decays exponentially to zero as

$$(x_t)_{\mathcal{D}_{jd}} \sim \mathcal{N}(0, f_d^{2(t-jd)}(\Sigma)_{\mathcal{D}_{jd}})$$

We assume that this has approximately decayed to zero much before $t = (j+1)d$.

For every d frames, at $t = jd$, the values of x_t along the new indices, \mathcal{A}_t , become nonzero and the values of x_t along existing indices, \mathcal{D}_t , start to decay exponentially. The values of x_t along all other existing indices follow an independent AR-1 model. After a short period Δ_d which is much less than d , the variances of x_t along the new indices increase to some stable values and the values of x_t along the decaying indices decayed to zero. Therefore, N_t is piecewise constant from $t = jd + \Delta_d$ to $t = jd + d$.

REFERENCES

- [1] C. Qiu and N. Vaswani, "Real-time robust principal components' pursuit", in *Allerton Conference on Communication, Control, and Computing*, 2010.
- [2] C. Qiu and N. Vaswani, "Support-predicted modified-cs for principal components' pursuit", in *IEEE Intl. Symp. on Information Theory (ISIT)*, 2011.
- [3] Jolliffe I.T., *Principal Component Analysis*, Springer, second edition, 2002.
- [4] M. Brand, "Incremental singular value decomposition of uncertain data with missing values", in *European Conference on Computer Vision*, 2002, pp. 707–720.
- [5] Sam Roweis, "Em algorithms for pca and spca", *Advances in Neural Information Processing Systems*, pp. 626–632, 1998.
- [6] Fernando De La Torre and Michael J. Black, "A framework for robust subspace learning", *International Journal of Computer Vision*, vol. 54, pp. 117–142, 2003.
- [7] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?", *submitted to Journal of the ACM, December 2009*.
- [8] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Sparse and low-rank matrix decompositions", in *Allerton Conference on Communication, Control, and Computing*, 2009.
- [9] D. Skocaj and A. Leonardis, "Weighted and robust incremental method for subspace learning", in *IEEE Intl. Conf. on Computer Vision (ICCV)*, Oct 2003, pp. 1494–1501.
- [10] Yongmin Li, Li qun Xu, Jason Morphet, and Richard Jacobs, "An integrated algorithm of incremental and robust pca", in *IEEE Intl. Conf. Image Proc. (ICIP)*, 2003, pp. 245–248.
- [11] John Wright and Yi Ma, "Dense error correction via l1-minimization", *IEEE Trans. on Info. Th.*, vol. 56, no. 7, pp. 3540–3560, 2010.
- [12] E. Candes, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements", *Communications on Pure and Applied Mathematics*, vol. 59(8), pp. 1207–1223, August 2006.
- [13] Joel A. Tropp, "Just relax: Convex programming methods for identifying sparse signals in noise", *IEEE Trans. on Info. Th.*, vol. 52, no. 3, March 2006.
- [14] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders, "Atomic decomposition by basis pursuit", *SIAM Journal on Scientific Computing*, vol. 20, pp. 33–61, 1998.
- [15] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction", *IEEE Trans. Info. Th.*, vol. 55, no. 5, pp. 2230–2249, May 2009.
- [16] D. Needell and J.A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples", *Appl. Comp. Harm. Anal.*, vol. 26, pp. 301–321, 2008.
- [17] Y. Wang and W. Yin., "Sparse signal reconstruction via iterative support detection", *SIAM Journal on Imaging Sciences*, pp. 462–491, 2010.
- [18] Richard G. Baraniuk, Volkan Cevher, Marco F. Duarte, and Chinmay Hegde, "Model-based compressive sensing", *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1982–2001, 2010.
- [19] N. Vaswani, "Ls-cs-residual (ls-cs): Compressive sensing on least squares residual", *IEEE Trans. Sig. Proc.*, vol. 58(8), pp. 4108–4120, 2010.
- [20] N. Vaswani and W. Lu, "Modified-cs: Modifying compressive sensing for problems with partially known support", *IEEE Trans. Signal Processing*, September 2010.
- [21] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information", *IEEE Trans. Info. Th.*, vol. 52(2), pp. 489–509, February 2006.
- [22] D. Donoho, "Compressed sensing", *IEEE Trans. on Information Theory*, vol. 52(4), pp. 1289–1306, April 2006.
- [23] N. Vaswani and W. Lu, "Modified-cs: Modifying compressive sensing for problems with partially known support", *IEEE Trans. Signal Processing*, Sep 2010.
- [24] Arvind Ganesh, John Wright, Xiaodong Li, Emmanuel Candès, and Yi Ma, "Dense error correction for low-rank matrices via principal component pursuit", in *IEEE Intl. Symp. on Information Theory (ISIT)*, 2010.
- [25] E. Candes and T. Tao, "Decoding by linear programming", *IEEE Trans. Info. Th.*, vol. 51(12), pp. 4203–4215, Dec. 2005.
- [26] Y. Jin and B. Rao, "Algorithms for robust linear regression by exploiting the connection to sparse signal recovery", in *IEEE Intl. Conf. Acoustics, Speech, Sig. Proc. (ICASSP)*, 2010.
- [27] Kaushik Mitra, Ashok Veeraraghavan, and Rama Chellappa, "A robust regression using sparse learning for high dimensional parameter estimation problems", in *IEEE Intl. Conf. Acous. Speech. Sig.Proc.(ICASSP)*, 2010.
- [28] J.N. Laska, M.A. Davenport, and R.G. Baraniuk, "Exact signal recovery from sparsely corrupted measurements through the pursuit of justice", in *Asilomar Conf. on Sig. Sys. Comp.*, Nov 2009, pp. 1556–1560.
- [29] James E. Fowler, "Compressive-projection principal component analysis", *IEEE Transactions on Image Processing*, vol. 18, no. 10, pp. 2230–2242, Oct. 2009.
- [30] Kiryung Lee and Yoram Bresler, "Subspace-augmented music for joint sparse recovery", *arXiv:1004.3071v3*, 2011.
- [31] E. Candes and T. Tao, "The dantzig selector: statistical estimation when p is much larger than n", *Annals of Statistics*, 2006.
- [32] I. Daubechies, R. DeVore, M. Fornasier, and S. Gunturk, "Iteratively re-weighted least squares minimization: Proof of faster than linear rate for sparse recovery", in *Proceedings of the 42nd IEEE Annual Conference on Information Sciences and Systems (CISS 2008)*, March 2008.
- [33] E. Candes, "The restricted isometry property and its implications for compressed sensing", *Compte Rendus de l'Academie des Sciences, Paris, Serie I*, pp. 589–592, 2008.
- [34] H. Vincent Poor, *An Introduction to Signal Detection and Estimation*, Springer, second edition, 1994.
- [35] N. Vaswani, "Stability (over time) of modified-cs and ls-cs for recursive causal sparse reconstruction", *Allerton Conference on Communication, Control, and Computing*, 2010.
- [36] N. Vaswani, "Kalman filtered compressed sensing", in *IEEE Intl. Conf. Image Proc. (ICIP)*, 2008.
- [37] Junfeng Yang and Yin Zhang, "Alternating direction algorithms for l1 problems in compressive sensing", *Tech. Rep.*, Rice University, June 2010.
- [38] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix", *Tech. Rep.*, University of Illinois at Urbana-Champaign, August 2009.
- [39] Zhouchen Lin, Minming Chen, and Yi Ma, "Alternating direction algorithms for l1 problems in compressive sensing", *Tech. Rep.*, University of Illinois at Urbana-Champaign, November 2009.

SUPPLEMENTARY MATERIAL

The overall ReProCS algorithm is a recursive approach. In ReProCS(modCS) the sparse recovery algorithm, supp-pred-modCS, is itself recursive. Hence an important question is when and why will it be stable (error bounded by a time-invariant and small value)? Our simulations given in Sec. VI do indicate that it is stable. In this section, we provide the main arguments to show this analytically. These will be formalized in later work. *Everywhere in the discussion below, “bounded” means bounded by a time-invariant value.*

First consider supp-pred-modCS. We give here the main idea of an induction argument to show its stability. Suppose that at $t-1$, $\|S_{t-1} - \hat{S}_{t-1}\|_2$ is bounded and small and that the location error, $|p_{t-1} - \hat{p}_{t-1}|$ is also bounded and small. Since n_t is assumed to be bounded and small, the second induction assumption means that $|p_t - \hat{p}_{t-1}|$ is also bounded and small. It is easy to see then that the same holds for the support prediction errors $|\Delta_{t|t-1}| := |T_t \setminus \hat{T}_{t|t-1}|$ and $|\Delta_{e,t|t-1}| := |\hat{T}_{t|t-1} \setminus T_t|$. In fact, both $|\Delta_{t|t-1}|$ and $|\Delta_{e,t|t-1}|$ are bounded by $|p_t - \hat{p}_{t-1}|$. Using this and arguments similar to those in [35], we should be able to argue that the sparse recovery and support update step (modified-CS with Add-LS-Del) will also result in (i) $\hat{T}_{t|t}$ with bounded and small number of extras, $|\Delta_{e,t|t}|$, and misses, $|\Delta_{t|t}|$ and (ii) the error of the recovered sparse part, $\|S_t - \hat{S}_t\|_2$ being bounded and small. This step will need to assume that (a) the noise seen by modified-CS, β_t , is bounded and small; (b) A_t satisfies a certain RIP condition (if $|\Delta_{t|t-1}| \leq a$ and $|\Delta_{e,t|t-1}| \leq b$ then we will need $\delta_{2w+1+a+b} < (\sqrt{2} - 1)$) [33], [35]; and (c) most nonzero elements of S_t are large enough. In fact using this approach, the bound on the final number of extras, $|\Delta_{e,t|t}|$ will be zero. Thus, we will just need to argue that the final misses, $\Delta_{t|t} := T_t \setminus \hat{T}_{t|t}$, will result in bounded and small centroid observation error, $\omega_t := p_{t,\text{obs}} - p_t$. We show how to do this in the next paragraph. This, along with ensuring the stability of $\Sigma_{t|t}$, will ensure bounded and small $|p_t - \hat{p}_{t|t}|$. *Note that we use a KF in this paper, but in general the above argument will go through with any stable linear observer.*

Let $T := \hat{T}_{t|t}$ and $\Delta := T_t \setminus T$. To obtain a bound on $|\omega_t|$, notice that $p_t = \frac{1}{|T_t|} \sum_{i \in T_t} i$ while $p_{t,\text{obs}} = \frac{1}{|T|} \sum_{i \in T} i$. Since the final number of extras is zero, $T_t = T \cup \Delta$ and so $|T_t| = |T| + |\Delta|$. Thus, $(|T| + |\Delta|)p_t - |T|p_{t,\text{obs}} = \sum_{i \in \Delta} i$ and so $|T|(p_t - p_{t,\text{obs}}) = \sum_{i \in \Delta} (i - p_t)$. Using (12), $|T_t| = 2w + 1$ and $|(i - p_t)| \leq w$ for all $i \in T_t$. Thus,

$$|\omega_t| = |p_t - p_{t,\text{obs}}| \leq \frac{|\Delta_{t|t}|w}{2w + 1 - |\Delta_{t|t}|} \quad (28)$$

In practice, if there are extras (thresholds are not set high enough to ensure zero extras), then we can show that

$$|\omega_t| \leq |\Delta_{t|t}| \frac{w}{2w + 1 - |\Delta_{t|t}|} + |\Delta_{e,t|t}| \frac{\max_{j \in \Delta_{e,t|t}} |j - p_t|}{2w} \quad (29)$$

The above arguments, once formalized, will show that $S_t - \hat{S}_t$ is bounded and small. Since $\hat{L}_t - L_t = S_t - \hat{S}_t$, the same will hold for $\hat{L}_t - L_t$. But the above arguments require assuming that the “noise” seen by modCS, $\beta_t = \hat{P}'_{t,\perp} L_t$ is bounded and small. To show this we will need to show that our recursive

PCA algorithm given in the Sec V is accurate enough, even when we use \hat{L}_t instead of L_t . Also, we will need to assume that each element of x_t (and hence of L_t) is bounded, e.g., follows a truncated Gaussian or uniform distribution.